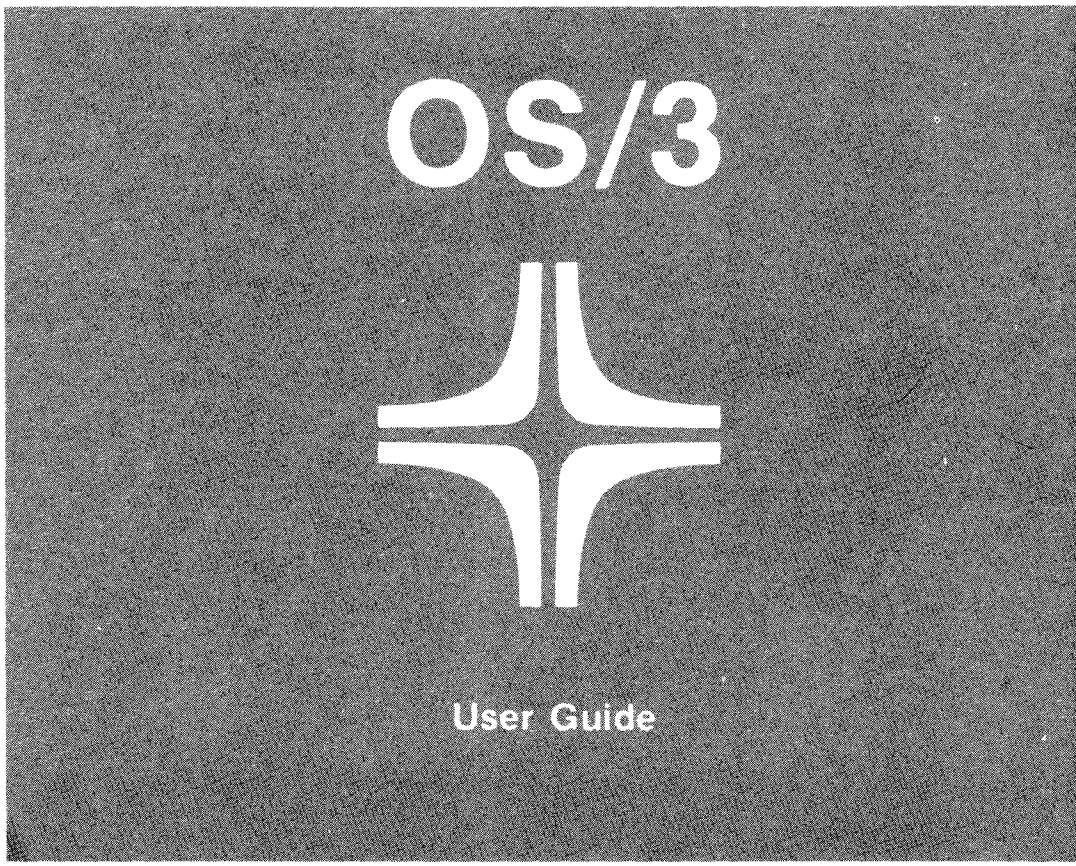
Integrated Communications Access Method (ICAM)

Standard MCP Interface (STDMCP)



SPERRY UNIVAC

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-8550 Rev. 5
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Preface	1 thru 3	
Contents	1 thru 9	
1	1 thru 9	
2	1 thru 193	
3	1 thru 102	
Index	1 thru 7	
User Comment Sheet		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (➔) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (➔) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

The standard interface is a logical interface that provides a general communications capability with message queueing and a message processing capability.

This user guide provides all of the macroinstructions, programming requirements, and terminal information you need to interface with the standard interface.

You will need this user guide only if you are writing your own communications program. Programs that deal with the standard interface must be coded in basic assembly language (BAL); therefore, you must ensure your system includes the OS/3 assembler if you write your own program. Specifically, it describes the ICAM standard interface macroinstructions you use when you write BAL applications programs that interface with ICAM networks. The intended audience is experienced BAL programmers who have a basic knowledge of communications data processing, and programmers whose experience is limited to systems other than SPERRY UNIVAC.

If you write your program in COBOL, you will require the COBOL message control system utility; you won't need this user guide because the utility converts your COBOL statements to instructions this interface recognizes for you.

This manual consists of three sections:

Section 1. Introduction

Provides an overview of ICAM standard interface processing. This description includes the overall configuration, components, and primary purpose.

Section 2. User Program Requirements

Describes the ICAM standard interface macroinstructions and how the user can write a communications program using these unique macroinstructions within his application program. Also describes the control packets, error conditions, parameters, and work areas necessary to this interface.

Section 3. Remote Terminal Characteristics

Describes the characteristics of the specific terminals that ICAM supports.

This manual is one of a series designed to guide you in programming and using the SPERRY UNIVAC Operating System/3 (OS/3) integrated communications access method (ICAM). Depending on your need, you may want to refer to the current version of one of the other ICAM manuals. Complete manual names, their ordering numbers, and a general description of their contents and use is as follows:

- Integrated Communications Access Method (ICAM) Concepts and Facilities, UP-8194

  Provides an overview of the facilities offered by ICAM including the hardware supported, the types of programs supported (assembler, COBOL, and RPG II), and the services provided (polling, queueing, buffering, etc).

- ICAM Network Definition and Operations User Guide, UP-8947

  Describes how to define an ICAM network, submit it to the system generation procedure, and load and operate the resulting ICAM symbiont. Many sample network definitions are provided to make it easier to define your ICAM network. In addition, most of the required *hands on* functions are described here. These functions include loading ICAM, establishing a dynamic session from the terminal, communicating with ICAM, etc.

- ICAM Transaction Control Interface (TCI) User Guide, UP-8551

  The transaction control interface is designed for applications that are transaction oriented, i.e., an inquiry followed by a response. The interface is primarily devoted to supporting the SPERRY UNIVAC Information Management System (IMS). If you are using IMS, you do not need this user guide, because IMS automatically converts requests you make in your IMS action programs to transaction control statements. You may write your own communications program to work with this interface. However, if you do, you will need this user guide because it contains the macroinstructions, programming requirements, and terminal characteristics you need in your program. Note that user programs that work directly with the transaction control interface must be coded in basic assembly language and your system must include the OS/3 assembler.

- ICAM Direct Data Interface (DDI) User Guide, UP-8549

  The direct data interface commonly supports ICAM utility programs and programs written in the RPG II language. If you are using an ICAM utility only, or your program is written in RPG II, you won't need this user guide because the utility programs and the RPG II compiler automatically convert any requests by your program to the proper instructions needed to work with this interface.

  The direct data interface also enables you to write your own communications program to work with it. If you do this, you must take care of your own message buffering and queueing, thus saving considerable main storage overhead. If you write a program to work with direct data interface, it must be written in basic assembly language, and your system must include the OS/3 assembler.

- ICAM Communications Physical Interface (CPI) User Guide, UP-8945

  The CPI interface requires the least amount of main storage, but it also provides a minimum amount of support. If you use this interface, you must have considerable knowledge of data communications because your program must initialize the hardware, format all output messages using the appropriate protocol, perform any required translations, acknowledge and process all input messages, and perform all error detection and recovery procedures. In addition, your program must be written in basic assembly language; therefore, your system must include the OS/3 assembler.

- ICAM Utilities User Guide, UP-8552

  Describes the utilities provided by ICAM. These utilities:

  1. Enable your processor to emulate a SPERRY UNIVAC 1004 Card Processor System
  2. Enable your processor to operate as a remote job entry/batch terminal to a SPERRY UNIVAC 1100 System
  3. Provide a facility to enable you to submit batch jobs from a remote terminal
  4. Provide the capability to produce printed reports from journal files
  5. Supply the software to create a module that converts communications requests in your COBOL program to instructions recognizable by the ICAM standard interface
  6. Describe how to run RPG II under ICAM as a utility
  7. Describe a routine to dump the single line communications adapter (SLCA)

- ICAM Message Processing Procedure Specification (MPPS) User Guide, UP-8946

  MPPS enables you to write message processing routines and include them in your ICAM network. This makes it possible for ICAM to analyze and process input messages before they are made available to your program, including the establishment of priority based on message content. Message processing routines can also be used to process output messages, including rerouting, if necessary, due to hardware and software error conditions.

  You do not need to include message processing routines in your network – they are totally optional; hence, your need for this user guide depends on your requirements.

- ICAM Programmer Reference, UP-8269

  This reference summarizes the information found in the other ICAM manuals. No introductory information or examples are given; however, it is a useful document when you are familiar with ICAM and you need a quick reference to macroinstructions, formats, and tables.

Contents

3. REMOTE TERMINAL CHARACTERISTICS

TABLES

1. Introduction

1.1. GENERAL

The standard interface (STDMCP) provides the software support you need to write BAL programs that access communications networks. Standard interface user programs have little control over the networks; they are isolated from hardware, line protocol, and message format dependencies. Thus, you need only a basic knowledge of communications to write them. You send and receive messages to and from network buffers via message queues in much the same way as you, in a noncommunications environment, read and write records to and from files stored on peripheral devices. You pass messages between work areas in your standard interface programs and selected queues.

Figure 1-1 illustrates the transfer of messages between terminals and programs in a standard interface system. Software elements called remote device handlers (RDHs) take messages from the buffers associated with a line and place them in queues in a form acceptable to your program. Similarly, the RDHs take messages sent to queues by your program and place them in line buffers in a form acceptable to communications adapters.

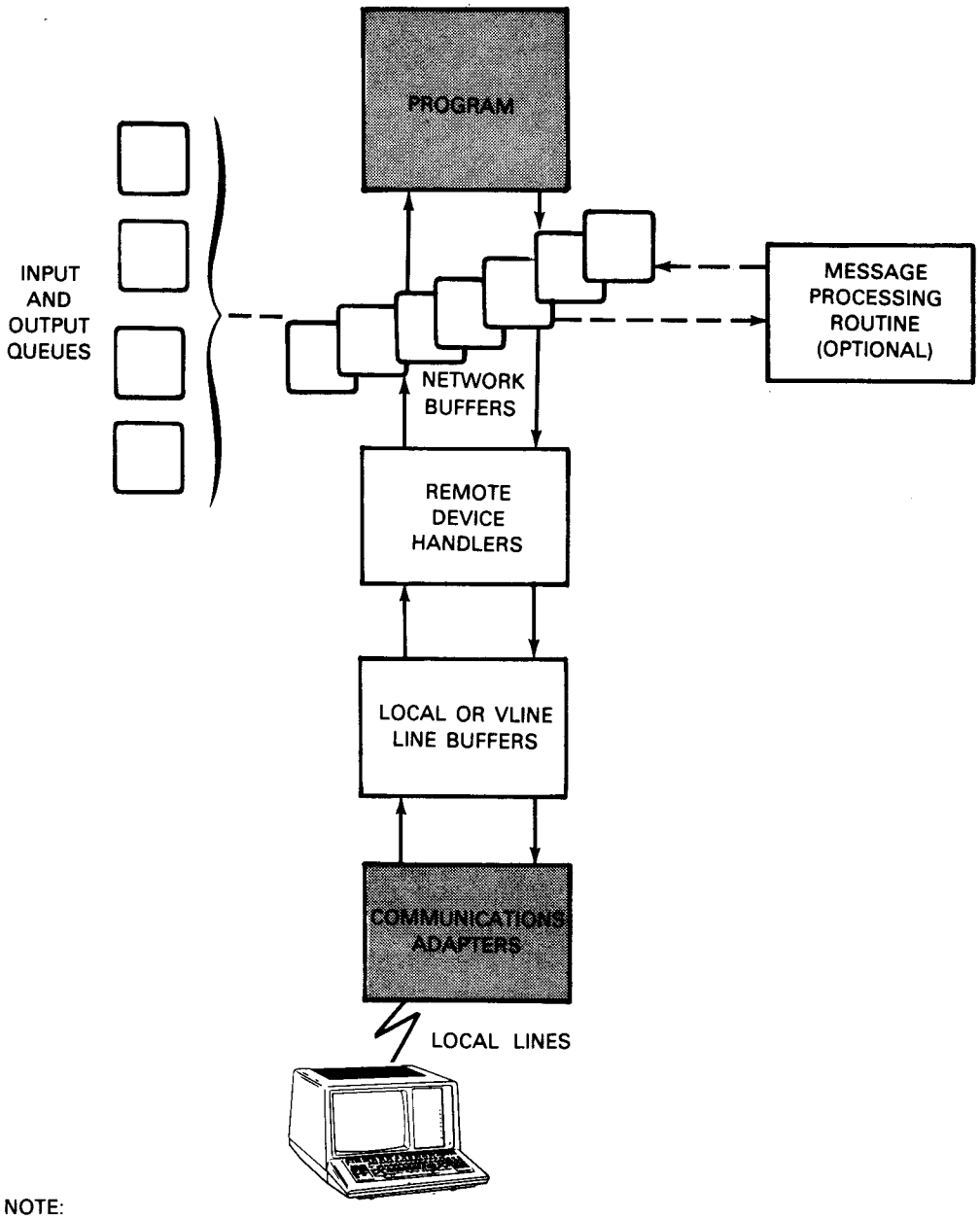
1.2. ICAM INTERNAL ELEMENTS

ICAM has a modular structure consisting of the elements described in the following subsections.

1.2.1. Communications Control Area

The communications control area contains the tables required to define and control a communications network. (A communications network is merely the definition of all of the lines, terminals, and queues, etc that you have assembled using the macroinstructions defined in the ICAM network definition and operations user guide/programmer reference, UP-8947 (current version).

A network definition may also include message processing procedure specifications (MPPS). A communications network definition may consist of one or more communication lines and their associated terminals. There are two types of networks: dedicated and global. The resources specified in a dedicated network become available only to the program that calls that dedicated network. With global networks, all defined resources are assigned to ICAM and many programs (both in the local computer and in a related remote computer) can share resources at the same time.



NOTE:

Shaded areas are not part of ICAM.

Figure 1-1. Standard Interface Communications System

After ICAM is loaded and initialized, your program must execute a network request (NETREQ) imperative macroinstruction if a dedicated network is being used. This assigns the specified dedicated network to your program and permits it to execute all other communications macroinstructions until it is ready to release the network. If you are using a global network, your program requests attachment to the global networks via an NATTACH macro.

Your program terminates a communication operation by executing the network release (NETREL or NDETACH) macroinstruction. If a program terminates, either normally or abnormally, without executing the NETREL/NDETACH macro, ICAM automatically performs the network release functions for you.

1.2.2. Activity Control

The activity control module is a central function of ICAM that monitors and schedules all communication activities within the ICAM system. All processing within ICAM is initiated by activity control, and all exits by ICAM functional elements are via the activity control module.

1.2.3. Remote Device Handlers

A family of remote device handlers (RDHs) provides the logic and control required to interface the characteristics of specific devices to other ICAM elements. Each RDH provides the following functions:

- Text translation
- Conversion of device independent control expressions (DICE) sequences to native hardware codes
- Optional conversion of native hardware codes to DICE sequences
- Poll qualification and polling where applicable
- Insertion of message framing and address characters on outgoing messages
- Removal of framing and address characters on input messages
- Retransmission control of messages when errors are detected
- Accumulation of network statistics regarding line and terminal operation

There are two types of buffers that are processed by an RDH. They are network buffers and line buffers. Network buffers store messages in a format suitable for internal processing by other ICAM elements or by user programs.

Line buffers contain message text, with framing characters, in a format that is native to a particular type of terminal or device. On outgoing messages, the RDH inserts the proper framing characters into the line buffer and moves the text, after translation, from the network buffers to the line buffers. The reverse procedure is performed for incoming messages. In this case, framing characters are removed and the text characters are translated and moved to the network buffers.

1.2.4. Deferred User Service Task

Deferred user service task (DUST) routines provide all functions for ICAM that are not time critical and are used infrequently. These routines support the following types of functions:

- ICAM load time initialization
- Network initialization at user network request
- Network termination
- Line request assignments and line release requests
- Communication subsystem initialization and parameter loading
- Line connection and automatic dialing
- Console operator communication functions
- Mapping of physical communication lines to logical lines
- Program termination

1.2.5. Channel Control Routine

The channel control routine provides the physical interface to the communications subsystem hardware.

1.2.6. Message Queueing

Message queueing is the stacking of complete messages in main storage or in a specified disk file while they are waiting for service by an RDH or a program. A single queue consists of one or more messages with only header segments linked together. Message text that overflows the header segment is contained in additional segments that have secondary links out of their associated header segment.

A network may contain one or more message queues that are associated with lines, terminals, process files, or programs defined by a LOCAP statement. The logic of ICAM treats all queues as destination queues even if they are directed to programs.

1.2.7. Message User Service Task

A message user service task (MUST) routine provides a message staging service that isolates your program from the device dependencies associated with data communications. MUST is responsible for copying data into program work areas from the network buffer pool or copying data into the network buffers from program work areas. Network buffers are related to each dedicated network or to a global network.

1.2.8. Communications Network Controller

The communications network controller coordinates the flow of messages between the RDHs and the message queueing routine. In addition, it provides the interface to special message editing functions including the MPPS routines.

1.2.9. Message Processing Procedure Specifications

The message processing procedure specifications enable you to specify special processing functions that ICAM performs on input and output messages for your program without benefit of user routines. You select MPPS processing functions when you generate a network by coding a series of macroinstructions with their associated parameters.

1.3. GLOBAL NETWORKS

There are two kinds of networks available to you: dedicated and global. You can specify as many dedicated networks as you desire; however, whenever a dedicated network is assigned to a program, that network and its resources become unavailable to any other program until it is released. Dedicated networks deal only with resources residing in a single computer.

All resources in a global network, however, are assigned to a supporting task known as the global user service task (GUST). In addition, terminals, process files, and programs (defined by a LOCAP macroinstruction) become end users* that may be paired, thus enabling the exchange of data between them. The required resources that enable this data exchange are provided by GUST. The pairing of end users and the allocation of necessary resources is defined as a session. You should note that a global network does not belong to a single program; therefore, any legitimate program may request attachment to a global network at any time. (See 2.22.)

Global networks also permit multiple computers using ICAM to be tied together in a computer-to-computer global network environment. This enables the resources in one computer to be allocated (in session) with the resources in another computer in much the same way as in a single node environment. Of course, global network definitions must be resident in both computers and they must relate to each other. In this type of global environment, the computers are connected by a physical link called a VLINE. Each physical VLINE can in turn have up to 4096 logical channels.

1.4. MACROINSTRUCTION CONVENTIONS

ICAM provides a complement of macroinstructions to facilitate service requests between your program and ICAM. There are two types of macros used within ICAM: declarative and imperative.

Declarative macroinstructions generate nonexecutable code sequences in your program. They are used to allocate areas in main storage containing control information for various system services.

**Note that the term end user does not imply that the recipient of a message must have a processing capability – only that the end user is a legitimate destination for a message. Hence, a process file is a legal end user even though it cannot output or initiate a message like a program.*

Imperative macroinstructions generate executable code sequences in your program. These code sequences make up the interface between your program and the supervisor. Imperative macroinstructions are used to request services of the supervisor or to direct the operation of your program.

1.4.1. General Format of Macroinstructions

The general format of the macroinstruction is:

LABEL	△OPERATION△	OPERAND
symbolic-name	macro-mnemonic	parameters

A symbolic name can appear in the label field. It can have a maximum of eight characters and must begin with an alphabetic character.

The appropriate macroinstruction mnemonic must appear in the operation field and identifies the operation or service requested.

When parameters are specified in the operand field, they must be positional parameters and/or keyword parameters as required by the particular function.

Assembler rules regarding blank columns and continuation of the operand field must be followed.

Parameters must not be separated by blanks.

1.4.2. Positional Parameters

Positional parameters must be written in the order specified in the operand field and must be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission, except for the case of omitted trailing parameters.

Example:

Assume that LOADX is a supervisor macroinstruction with one mandatory positional parameter (phase-name) and three optional positional parameters (load-addr, error-addr, and R):

LABEL	△OPERATION△	OPERAND
[symbol]	LOADX	{phase-name / (1)} [,{load-addr / (Ø)}][,{error-addr / (r)}][,R]

Macroinstruction statements may be written:

```
1         10    16
          LOADX RECAPLØ5,INADDR,ERADDR,R
          LOADX RECAPLØ5,,ERADDR
          LOADX RECAPLØ5,INADDR
          LOADX RECAPLØ5
```

1.4.3. Keyword Parameters

A keyword parameter consists of a word or a code immediately followed by an equal sign, which is, in turn, followed by a specification. Keyword parameters can be written in any order in the operand field. Commas are required only to separate parameters.

Examples:

Assume that EXAMP is a macroinstruction with two mandatory keyword parameters (IOAREA1 and BLKSIZE) and eight optional keyword parameters (EODADDR, FORMAT, LACE, LBLK, SEQ, SIZE, UOS, and VERIFY):

LABEL	△OPERATION△	OPERAND
[symbol]	EXAMP	IOAREA1=symbol
		,BLKSIZE=n
		[,EODADDR=symbol]
		[,FORMAT=NO]
		[,LACE=n]
		[,LBLK=n]
		[,SEQ=YES]
		[,SIZE=n]
		[,UOS=n]
		[,VERIFY=YES]
		[MF=(E,parameter-list)]

Macroinstruction statements may be written:

```
1         10    16                                                    72
          EXAMP IOAREA1=WORKAREA,BLKSIZE=256,FORMAT=NO,EODADDR=ENDNAME, X
                SEQ=YES,SIZE=1,UOS=1,VERIFY=YES
          EXAMP EODADDR=ENDNAME,IOAREA1=1NAREA,UOS=1,BLKSIZE=128,       X
                MF=(E,SESS1)
```

1.4.4. Coding Conventions

The conventions used to delineate macroinstructions are:

- Capital letters, commas, parentheses, and equal signs must be coded exactly as shown.

Examples:

```
R
ALL
(1)
SIZE=
```

- Lowercase letters and words are generic terms representing information that you must supply. Such lowercase terms may contain hyphens and acronyms (for readability).

  Examples:

```
name
start-addr,end-adr
number-of-bytes
param-1
ccb-name
```

- Information contained within braces represents mandatory entries of which one must be chosen.

  Examples:

```
{PC }
{IT }
{AB }

{input-area}
{(1)       }
```

- Information contained within brackets represents optional entries that (depending upon program requirements) are included or omitted. Braces within brackets signify that one of the specified entries must be chosen if that parameter is to be included.

  Examples:

```
[,entry-number]
[,R]
[,{ccb-name}]
[ {ALL     } ]
[ {(1)     } ]
[,ERROR=symbol]
[,WAIT=YES]
```

- An ellipsis (three periods) indicates the omission of a number of obvious entries.

  Example:

```
ccb-name-1....,ccb-name-n
```

- A positional parameter may consist of a list of parameters called subparameters, which are separated by commas. If a subparameter is omitted, the comma must be retained, except for the case of trailing subparameters.

  Example:

  ```
  {entry-point,save-area,input-area,length}
  {(1)                                    }
  ```

- An optional parameter that has a list of optional entries may have a default specification that is supplied by the operating system when you have not specified the parameter. Although you may specify the default with no adverse effect, it is considered inefficient to do so. For easy reference, when a default specification occurs in the format delineation, it is shaded.

  Example:

2. User Program Requirements

2.1. GENERAL

In an OS/3 system that uses ICAM, most output messages originate from user programs. User programs also are the final destination for most input messages from remote terminals, unless you allow the message processing procedure specification (MPPS) routines to perform automatic message switching. (For details, see the message processing procedure specification user guide, UP-8946 (current version).)

To send a message to a remote terminal or obtain a message from ICAM, your program must follow certain rules and procedures. This section explains and defines the requirements that relate directly to your program, i.e., the statements it issues, the interface areas (DTFs) it sets up, and the way it controls its environment.

Your program's interface to ICAM is a set of procedures called the standard interface (STDMCP).

2.2. GETCP/PUTCP MACROINSTRUCTIONS

Two basic functions are performed by ICAM's standard interface:

- a Get function to access input messages received by ICAM from communications devices; and
- a Put function to transfer messages to ICAM for transmission to communications devices.

Thus, the standard interface is sometimes called the Get/Put interface. The Get function is initiated in your program by the execution of the GETCP imperative macroinstruction; the Put function is initiated by the execution of a PUTCP macroinstruction.

In addition to the GETCP and PUTCP macroinstructions, your program needs two additional elements to process messages: a user work area and DTFCP file tables.

2.3. USER WORK AREA

A user work area is a storage area in your program that holds messages you have received in response to a GETCP request, or that you will send via a PUTCP request. You must use two bytes preceding the first message text character to insert a count of the number of characters to get from the input message, or a count of the number of characters in the output message. This two-byte field, known as the work area prefix, must be aligned on a half-word boundary. The byte count stored is the length of the message text, not including the prefix length.

When you execute a GETCP macroinstruction, you use the work area prefix to specify the maximum number of characters that can be transferred to the program work area referenced by the GETCP. When control is returned to your program from a GETCP, the work area prefix contains a count of the number of bytes actually transferred to the work area.

Similarly, when you execute a PUTCP macroinstruction, you use the work area prefix to specify the total byte count to be sent from the user work area.

Figure 2-1 shows the format of a typical message work area including the required work area prefix. Remember that when you issue a GETCP or PUTCP macroinstruction, the address of the work area referenced is that of the first byte of the work area prefix, not the beginning of the data to be sent or received. Also remember that the value specified in the length field is that of the data field, not including the length field itself.

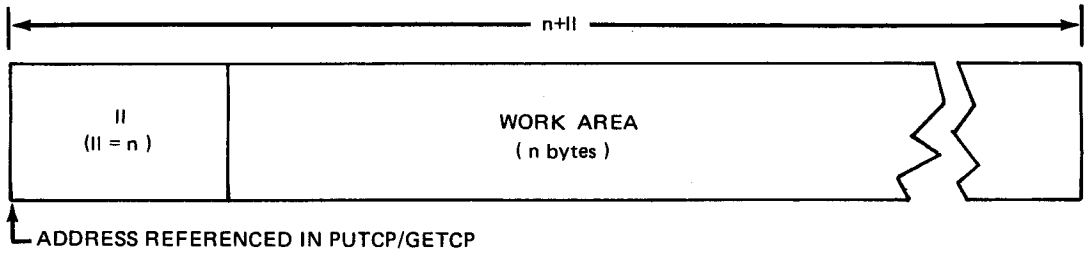


Figure 2-1. User Message Work Area Format

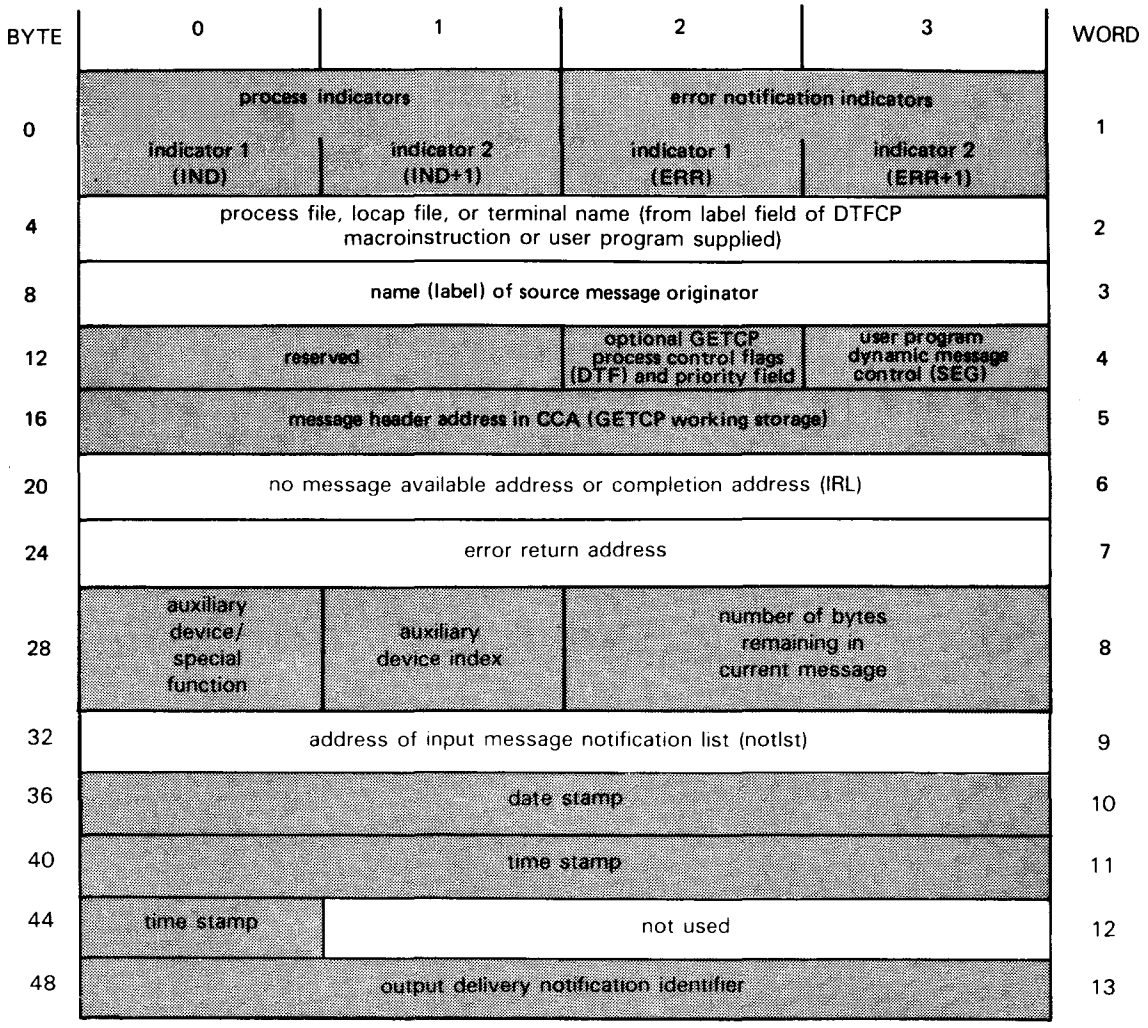
2.4. DTFCP FILE TABLES

The DTFCP file tables contain identification and control information applicable to the Get or Put function to be performed. They are used by your program and ICAM. DTFCP file tables are generated by the declarative macroinstruction DTFCP, TYPE=GT, or TYPE=PT.

Your program can modify various fields within the file tables to select the various message control options available. For instance, after completing a GETCP or PUTCP request, your program can check error and status information placed in the file tables by ICAM.

Figures 2-2 and 2-3 describe input and output DTFCP file tables. A brief description of the file table's fields follows each figure. For detailed information about the use of each field in the file tables, see the DTFCP macroinstruction descriptions in 2.21.1 and 2.21.2.

2.4.1. Input DTFCP File Table



NOTE:

Shaded areas are system-supplied parameters; clear areas are user-program-supplied parameters.

Figure 2-2. Input DTFCP File Table Field Descriptions

- Word 1
  - Field 1 - Process indicators/IRL flag

    A field divided into two single-byte, bit-oriented indicators that reports the status of the current message, such as message complete or more data available.

    It contains a bit setting for optional immediate return line (IRL). If selected, word 6 becomes a completion address and the GETCP is deferred. You set flags to initiate IRL, while ICAM sets status flags for you.

  - Field 2 - Error notification indicators

    If TM#PIER is set in field 1, look in these bytes for error details. This field is divided into two single-byte, bit-oriented indicators and is primarily used for error notification.

- Word 2
  - The name you designate for the process file, locap file, or terminal
  - Input file name

  This can be either the label of the DTFCP macroinstruction that created this table or it can be supplied by the user program. However, it must match the label of the TERM, LOCAP, or PRCS macroinstruction in the network definition that defines the end user.

- Word 3

  The name of the source message originator. Although this normally is a terminal, the name isn't restricted to terminals.

- Word 4
  - Field 1 - Reserved for ICAM user
  - Field 2 - Optional GETCP process control flags (DTF) and priority field

    A bit-oriented field you can use at your option to control specific aspects of GETCP operations.

  - Field 3 - User program dynamic message control (SEG)

    A bit-oriented field that your program uses for dynamic control of messages.

- Word 5
  - Message header address in communications control area

A working storage address constant used by ICAM. You must not change this field.

- Word 6
  - No message available address or IRL completion address

    A user-supplied address used as a contingency address when a message is unavailable from a GETCP. This becomes the completion address field if you select the IRL option in word 1.

- Word 7
  - User program error return address

    A user program address used as an entry point to pass control to your program whenever an error occurs. If you do not specify an ERRET=address, the return follows the GETCP or IRL completion (inline).

- Word 8
  - Field 1 - Auxiliary device/special function

    A bit-oriented field that reports the status of an auxiliary device or handles special features of a terminal, such as bell received or function keys.

  - Field 2 - Auxiliary device index

    Field containing the auxiliary device index. Refer to the AUXn parameter of the TERM macroinstruction. (See the ICAM network definition and operations user guide, UP-8947 (current version).)

  - Field 3 - Number of bytes remaining in current message

    A count of the undelivered characters remaining in the current message. You must not change this field.

- Word 9
  - Input message notification list address

    The address of a table containing labels of process fields, locap files, or terminals, if you select the input notification option.

- Words 10, 11, 12
  - Date and time stamp

    Contains the date and time the message was first put into a queue, if you select the date/time stamp option.

- Word 13

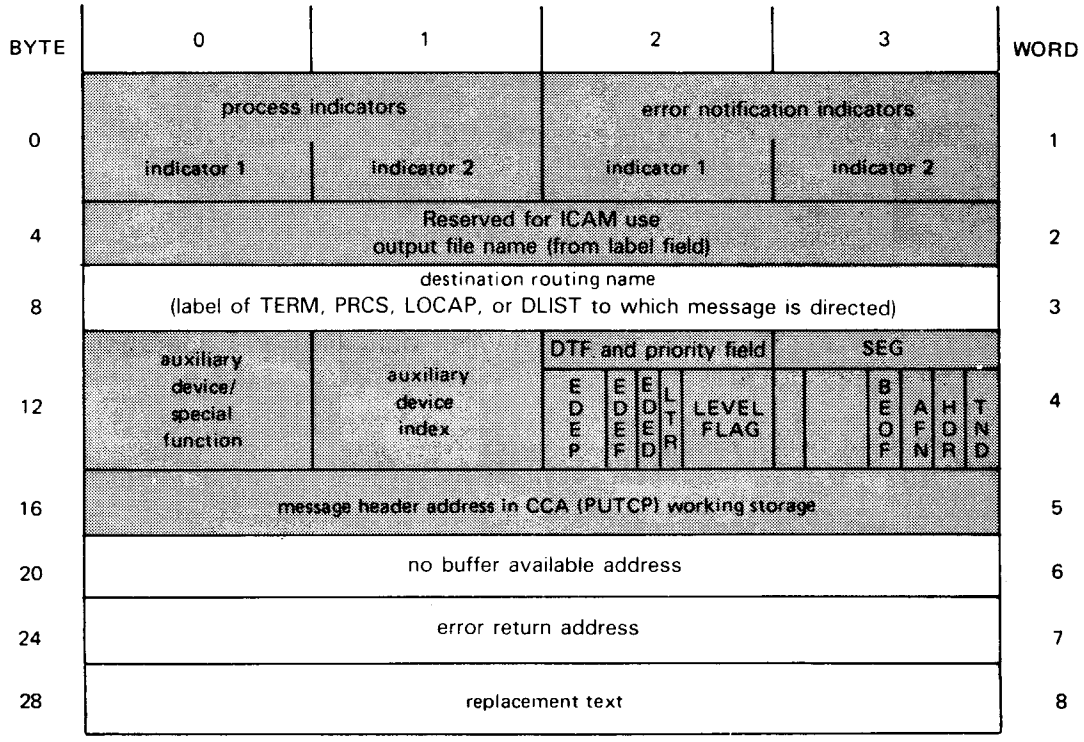
  - Output delivery notification information

Contains the output delivery notification identifier that was stored in the message if you selected the ODNR option when the message was output and you selected the ODN option on the GETCP. This information is only transferred to the DTFCP TYPE=GT file table if TM#PODN is set in field 2 of word 4 (TM#PDTF). Refer to 2.9 for a further description of ODN use with GETCP.

NOTE:

To select this option, the user must specify the ODN=YES parameter on the DTFCP macroinstruction. Otherwise, the user's code may be overwritten by ICAM.

2.4.2. Output DTFCP File Table



NOTE:

Shaded areas are system-supplied parameters; clear areas are user-program-supplied parameters.

Figure 2—3. Output DTFCP File Table Field Descriptions

- Word 1
  - Field 1 – process indicators/IRL flag

    A field divided into two single-byte, bit-oriented indicators that report the status of the current message.

    It contains a bit setting for optional immediate return line (IRL). If selected, word 6 becomes a completion address and the PUTCP is deferred.

    The field also contains an optional delivery notification setting. If selected, word 7 becomes a unique identifier.

    You cannot set IRL and output delivery notification at the same time. You set flags to initiate IRL and output delivery notification, while ICAM sets status flags for you.

  - Field 2 – Error notification indicators

    If TM#DIER is set in field 1, you look in these bytes for error details. This field is divided into two single-byte, bit-oriented indicators and is primarily used for error notification.

- Word 2
  - Output file name

    The label you specified for the DTFCP that created this table.

- Word 3
  - Destination routing name

    TERM, PRCS, LOCAP, or DLIST macroinstruction label to which the message is directed.

- Word 4
  - Field 1 – Auxiliary device/special function

    A bit-oriented field that controls an auxiliary device or handles special features of a terminal.

  - Field 2 – Auxiliary device index

    Field containing the auxiliary device index. Refer to the AUXn parameter of the TERM macroinstruction. (See the ICAM network definition and operations user guide, UP-8947 (current version).)

- Field 3 - Optional PUTCP process control flags (DTF) and priority field

  A bit-oriented field that you can use at your option to control specific aspects of PUTCP operations.

- Field 4 - User program dynamic message control (SEG)

  A bit-oriented field that your program uses for dynamic control of messages.

- Word 5

  - Message header address in communications control area

    A working storage address constant used by ICAM. You must not change this field.

- Word 6

  - No buffer available address or IRL completion address

    A user program address used as a contingency address when a buffer is unavailable for the first segment of an output message. This becomes the completion address field if you select the IRL option in word 1.

- Word 7

  - User program error return address

    A user program address used as an entry point to pass control to your program whenever an error occurs and you specified ERRET= address. If you do not specify an ERRET= address, the return follows the PUTCP (inline).

    This field becomes a user specified identifier field if you select output delivery notification.

- Word 8

  - A 4-byte field where you can put a DICE sequence to overlay the advancing information in the first four bytes of a previous segment that is on hold. This is used with the text replacement feature.

2.4.3. ICAM DSECTS

ICAM provides DSECTs (TM#PRCS and TM#DEST) that map the DTFCP file tables. They provide labels you can use for input and output message processing. The DSECTs are automatically generated in your program by the DTFCP macroinstruction. (See 2.21.1 and 2.21.2.)

2.5. QUEUES

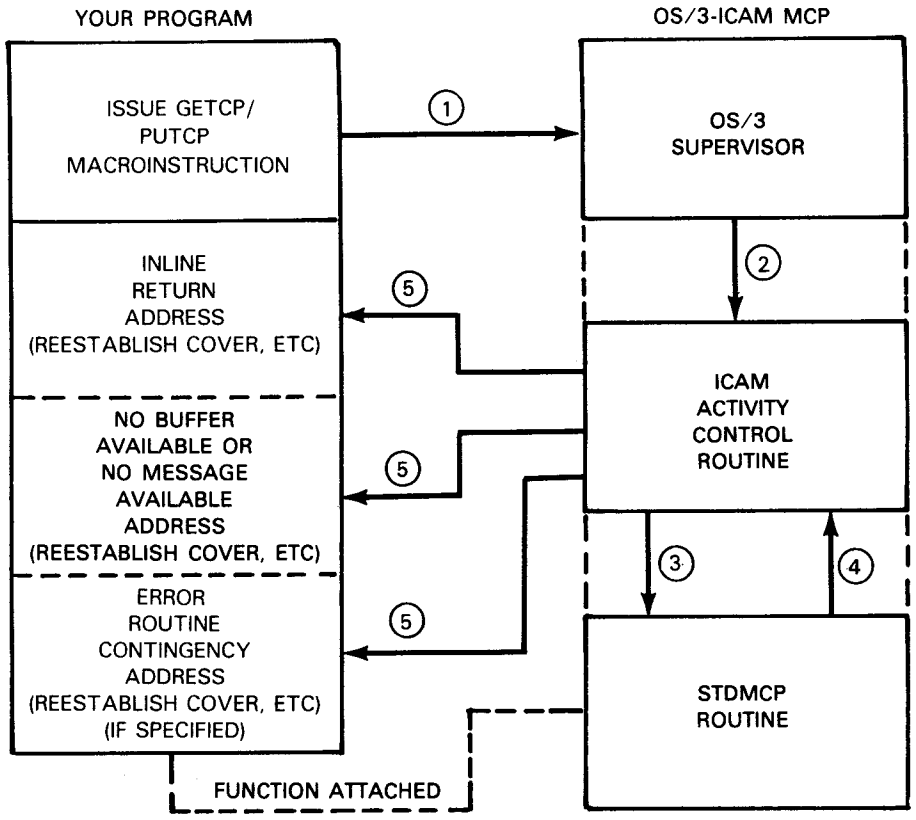
Your program sends and receives messages to and from buffers that are linked to a destination message queue. When you code a GETCP macroinstruction, you name the process file, input terminal queue, or locap file queue (global networks) that contains messages for your program. Similarly, when you code a PUTCP macroinstruction, you name the output terminal queue, process file, or locap file queue that is to receive your messages.

2.6. PROGRAM CONTROL INTERFACES

Normally, when you execute a GETCP or PUTCP macroinstruction, as shown in Figure 2–4, program control is transferred to the standard interface via the OS/3 supervisor and ICAM's activity control module. Upon successful completion of the request, ICAM returns control to your program inline, i.e., to the instruction following the GETCP or PUTCP macroinstruction. Your program may also specify two contingency return addresses in the DTFCP for each macroinstruction: an error return address for GETCP and PUTCP, and a no-message-available return address for GETCP or a no-buffer-available return address for PUTCP. Upon execution of a macroinstruction, the program releases control until successful completion of the function or until a contingency is detected. Program cover registers and saved information should always be reestablished when control is returned.

2.7. IMMEDIATE RETURN LINE USAGE

You can alter the normal Get/Put process by using the immediate return line (IRL) option, as shown in Figure 2–5. When you execute a GETCP or PUTCP request with IRL, control is transferred via the OS/3 supervisor to ICAM's activity control module. In this case, however, the GETCP or PUTCP request is queued for later processing, and control is immediately returned to your program's inline return address. Eventually, your program must execute a CYIELD macroinstruction or a GETCP or PUTCP without IRL to allow the standard interface to receive control to process the queued requests. With IRL, the DTFCP no-message-available or no-buffer-available contingency return address field must contain a completion address for return of control to the program when GETCP or PUTCP processing is completed.



NOTES:

(1) Program issues GETCP or PUTCP macroinstruction.

(2) OS/3 supervisor passes control to ICAM's activity control routine.

(3) ICAM activity control releases control to STDMCP to process the request.

(4) STDMCP processes request and returns control to the activity control routine.

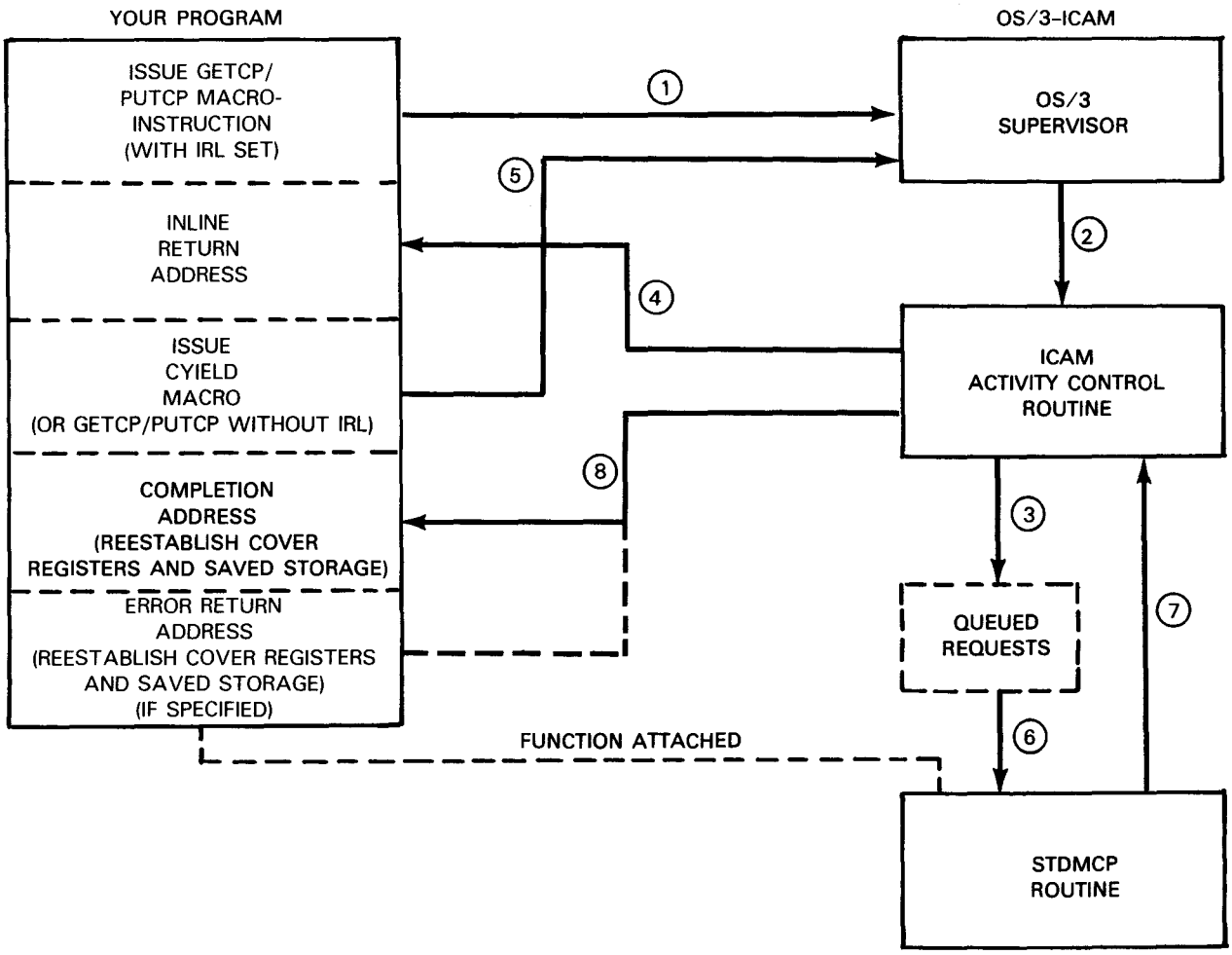
(5) Activity control schedules control to program as shown in the figure.

Figure 2—4. ICAM Normal GETCP/PUTCP Processing — IRL Not Set

You use labels provided by the TM#PRCS and TM#DEST DSECTs to initiate IRL. For Get processing, set TM#PIRL in the TM#PIND field; for Put processing, set TM#DIRL in the TM#DIND field.

When your program issues an IRL, it is logically disconnected from ICAM. You should be aware that your program is now line-event driven and not SVC driven; therefore, ICAM can't guarantee return registers other than registers 0 and 1. When you use IRL mode, keep all return areas and error return addresses (ERRET) under the same cover as all SVC calls to ICAM, if possible. Otherwise, you should reestablish cover at each return point.

You cannot use the IRL function in conjunction with the output delivery notification function.



NOTES:

(1) Program issues a GETCP or PUTCP macroinstruction.

(2) OS/3 supervisor passes control to ICAM activity control routine.

(3)(4) Because IRL is set, the request is queued and control is returned inline to your program.

(5) Program releases control by issuing a CYIELD macro.

(6)(7) All queued requests are processed by STDMCP. You are scheduled by the activity control routine, depending upon the results of each Get/Put.

(8) Control is returned to your program at its completion address or error address. At this point the cover registers and any storage containing saved information important to the user program must be reestablished.

Figure 2—5. ICAM Get/Put Processing with IRL

2.8. HOW TO USE DISTRIBUTION LISTS (DLIST)

A distribution list is a table of two or more destinations that you specify in your program or in a network definition. You use it to send the same message to all of the destinations listed in the table. Distribution lists are very useful because they save your program from having to send the same message to multiple destinations with individual PUTCP requests.

You specify distribution lists by means of DLIST macroinsructions either in your program or in the network definition. However, if you code all of your distribution lists in your program, you must code at least one DLIST macroinstruction in your network definition even if you don't plan to use it. This is because the COMMCT phase of ICAM system generation uses the DLIST macroinstruction call to include the necessary modules for distribution list processing.

To send a message to multiple destinations using a distribution list, you must:

1. Set field TM#DDTF in your output DTFCP to TM#DUDL if the distribution list is defined in your program. If the distribution list is defined in your network definition, no setting is required in TM#DDTF for distribution list processing. TM#DUDL must be reset.

2. For a network defined DLIST, move the label of the DLIST macroinstruction in the network definition into the TM#DENA field of the DTFCP. If your DLISTs are coded in your program, move the address of the associated DLIST in your user program into the TM#DENA field (rather than moving the label of the required DLIST in the network definition).

3. Set any other fields in the output DTFCP you require, for example, for a complete message.

4. Issue the PUTCP.

2.9. BATCH DEVICE CAPABILITY

ICAM offers full batch device support (DCT 2000, 1004 card processor, IBM 2780, and batch mode DCT 1000) but cannot offer restart capability if output fails after the first image is transferred. Once an image is removed from a queue, ICAM has no backup capability. You and ICAM treat each input card image and output print/punch image as a complete message. On the last input image, ICAM marks TM#PIND with TM#PBEOF to signify input batch end of file. On the last output image, you must mark TM#DSEG of the destination queue DTF with TM#DBEOF to signify output batch end of file. You must recognize the source name of the batch device and the start of file; ICAM marks EOF.

You can use IRLs and output delivery notification requests (ODNRs) with batch devices; however, ICAM can only restart where it left off in case of output line errors. If batch operation is used with main storage queueing, the BUFFERS macroinstruction in your network generation must specify that network buffers are at least 48 words in length or results will be unpredictable because the remote device handlers do not handle segmented images. If the message text length being output to a batch device is greater than the text area of a single network buffer, a no-buffer-available return is given.

2.10. OUTPUT DELIVERY NOTIFICATION REQUEST

The output delivery notification request feature, called ODNR, lets ICAM notify your program that a message it sent to a terminal was delivered. If ICAM cannot deliver a message, it reports status to your program to explain why delivery was impossible. ICAM holds undeliverable messages on queue until the terminal is able to receive them or your program cancels them (see the QCLEAR macroinstruction).

The ODNR feature applies to terminals only, that is, you cannot use it to report delivery of a message sent to a process file or to a locap file. However, terminals may be located locally in this computer node or in a remote computer node.

To incorporate ODNR into your program:

1. Set TM#DODNR in field TM#DIND of the output DTFCP.

2. Move any nonzero 4-byte identifier (such as a message number) into field TM#DERA of the output DTFCP.

3. Issue the PUTCP.

At this time, normal PUTCP status (did I issue a valid PUTCP request, and was it accepted by ICAM for processing?) is made available to your program in the status indicators of the DTFCP. Note, if the PUTCP is invalid, your program does not receive control at the DTFCP ERRET address because this field (TM#DERA) already contains the 4-byte identifier you want returned when the message is delivered; therefore, return is always inline unless there is no buffer available.

When the message is delivered (or ICAM is unable to deliver it), your program receives control at the address you specified in the ERRET operand of the NETREQ/NATTACH macroinstruction. ODNR status is now available in the leftmost byte of register 0 (byte 0), and the nonzero 4-byte identifier is in register 1. At this time, your program should test register 0 to find out why it got control at its NETREQ/NATTACH ERRET address. You need to do this because your program may receive control due to a DUST error or a line-down condition as well as an ODNR return. If byte zero of register 0 is not zero, the return is an ODNR return. After you finish analyzing byte zero, you should test register 1 to identify the particular message.

ICAM reports unsuccessful status only after it exhausts its retry count. If successful input is received from a terminal marked down, output is retried and a successful delivery notice may follow. An unsuccessful report may also be repeated if the terminal can only send good input. Line status conditions you may receive are described in Table 2-1.

When a message is rerouted to a process file or locap file queue, the ODN identifier may also be retrieved for a message on subsequent GETCPs if you:

1. specify the ODN=YES operand on the input DTFCP macroinstruction;

2. set TM#PODN in field TM#PDTF of the input DTFCP; and

3. issue the GETCP.

At this time, the ODN identifier that was stored in the message when it was output is moved to the field TM#PODNN of the input DTFCP. This identifier may then be retrieved by the user program to uniquely identify the message that has been received.

NOTES:

1. *You must not use ODNR with the immediate return line option (IRL).*

2. *When your program receives control at its NETREQ/NATTACH error return (ERRET) address, the return may be due to an ODNR notice, a line-down condition, or, when you don't specify DUSTERR=INLINE, a DUST macroinstruction error. We suggest you determine the kind of return by saving register 0 in a work area and testing as follows:*

   - *Test byte 0 - All ODNR returns are posted in byte 0.*
   - *Test byte 1 - All line-down conditions are posted in byte 1.*
   - *Test bytes 2 and 3 - All DUST macroinstruction errors are reported in bytes 2 and 3.*

3. *If a line or terminal is down at the time your program issues a PUTCP, or goes down before ICAM attempts to send the message, ODNR is not reported.*

4. *If a hardware error occurs on a line while ICAM is sending a message, a line-down condition notice replaces ODNR.*

5. *If the last or only terminal on a line goes down due to a protocol error while ICAM is sending a message, both ODNR and line-down condition notices are reported. ODNR is provided first.*

Table 2-1. Output Delivery Notice Status Codes

Output Completion-Status			STDMCP Byte 0 of Register 0
Successful	UNISCOPE, UTS 400, or DCT 1000		TM#DNNEM
	DCT 1000 MSG truncated		
	TTY, DCT 500/524/475/UTS 10		
Device down	UNISCOPE auxiliary device status 0. Lost or no status.		TM#DNNAX
	UNISCOPE auxiliary device status 1. Ready status but the tape cassette system (TCS) function is inoperative.		TM#DNNAX ++ TM#DAUX1
	UNISCOPE auxiliary device status 2. Out of paper on the communications output printer (COP) or 800 terminal printer (TP). End of tape on TCS.		TM#DNNAX ++ TM#DAUX2
	UNISCOPE auxiliary device status 3. Data error on the TCS. Several attempts at backward one block and repeat of the TCS function have been made by the RDH. The number of attempts by ICAM at error recovery is determined by the value specified by the LINE macroinstruction of the CCA. The default value is 4.		TM#DNNAX ++ TM#DAUX3
	UNISCOPE auxiliary device status 4. Device is not responding. It may be disconnected or a read of unwritten tape may have occurred.		TM#DNNAX ++ TM#DAUX4
	TTY, DCT 500, DCT 524, or DCT 475 UTS 10	Error during output to auxiliary device	TM#DNNAX
		BREAK received during TTY mode tape output read command	TM#DNNAX ++ TM#DNBRK
Abort output	This status is received from 1004.		TM#DNOAB

2.11. LINE-DOWN NOTIFICATION

If you are using a dedicated network and a line goes down, your program receives control at the address you specified in the ERRET operand of the NETREQ macroinstruction. Assuming the line down is a temporary condition, such as an accidental disconnection, a faulty telephone line, or a bad telephone connection, you can reactivate the line by first releasing it, then reactivating it, i.e., by issuing a LNEREL macroinstruction followed by a LNEREQ macroinstruction for the downed line.

You may issue as many DUST service requests as you want. Your program always receives control inline following each service request when the function is complete. (Unless the service function cannot be performed and you don't specify DUSTERR=INLINE. In this case, you receive control at the normal NETREQ macroinstruction error return address.) While your service request is being processed, ICAM continues to send messages and queue messages it receives. You may have deferred GETCPs pending against some of these queues; these remain active, including those for the downed line. You must not reissue these deferred Gets. ICAM will not give your program control at one of these return addresses until your program issues a CYIELD macroinstruction. When your program issues the CYIELD macroinstruction, ICAM gives your program control at the next return address it has queued. For example, a message available address, another line-down error address, etc.

When your program receives control at the ERRET address, the 4-character name of the line that is down is in register 1, and one of three error codes is in byte 1 of register 0. The error codes are described in Table 2-2.

If you are using a category H packet switched public data network and line status is reported, register 1 contains the 1-4 character name of the remote locap file, and register 0 contains the permanent virtual circuit line status.

Table 2—2. Line-Down Notification Error Codes

Error Condition	Comments	Byte 1 of Register 0
Line down	Line disconnected (loss of DSR)	TM#DNLNO
Undefined terminal	An undefined terminal is responding to poll	TM#DNSIT
Terminal down	All terminals on the line are down, but ICAM is still connected. An all-terminals-down message may also be sent to the system operator. If any terminal on the line responds with good input, the line is marked up. You may activate the line with an unsolicited console command.	TM#DNDNA

If the error code is TM#DNSIT, a message is sent to the system operator to warn him of the situation. You should issue an immediate line release for security reasons.

If you are using a global network, line-down conditions are not reported to your program. They are handled by the global user service task program (GUST).

When your program receives control at its NETREQ/NATTACH error return (ERRET) address, the return may be due to an ODNR notice, a line-down condition, a DUST macroinstruction error (when you don't specify DUSTERR=INLINE), or notification of a change in permanent virtual circuit (pvc) line status if you are using a category H packet

switched public data network. Determine the kind of return by saving register 0 in a work area and testing as follows:

- Test byte 0 – All ODNR returns are posted in byte 0.
- Test byte 1 – All line-down conditions are posted in byte 1.
- Test bytes 2 and 3 – All DUST macroinstruction errors are reported in bytes 2 and 3.

This includes category H public data network permanent virtual circuit (pvc) line status. See Table 2-3 for details.

Table 2-3. Public Data Network PVC Line Status

PVC Status	Hexadecimal Value
PVC down	42
PVC up	41

2.12. ERROR PROCESSING PROCEDURES

When an error occurs during the processing of an imperative macroinstruction, there are four types of problems that may have caused that error:

1. A bad Get or Put (for example, an illogical request to a nonexistent terminal or bad syntax).
2. A good Get or Put that couldn't be performed for some reason (for example, no network buffers available).
3. Some event occurred that caused the error (for example, a line-down condition or an unknown terminal on the line).
4. An error occurred during deferred user service task (DUST) macroinstruction processing.

Your program must have an error processing routine of some kind. We have revised error processing to make it simpler for you to determine what, if any, errors occurred during macroinstruction processing. Depending on whether you specify the DUSTERR operand of the CCA and LOCAP macroinstructions, ICAM reports errors and returns control to your program inline or at a specified error return address. (See the ICAM network definition and operations user guide, UP-8947 (current version) for details on coding the DUSTERR parameter.)

If you have existing programs, you may decide not to specify DUSTERR=INLINE and code your programs for error return to an ERRET=address. However, we recommend that you take a simpler approach by specifying DUSTERR=INLINE and coding your programs for inline returns.

- DUSTERR=INLINE Not Specified

  When you don't specify DUSTERR=INLINE and an error occurs, control is returned to the ERRET= address specified. Error return for the message processing functions (GETCP and PUTCP) is specified in the DTFCP macroinstructions. If no ERRET= address is specified, control is returned inline following the macroinstruction.

  For the deferred user service task (DUST) functions, error return is specified in the NETREQ or NATTACH macroinstruction. In some cases, DUST is unable to return to the error return address you specify. The following errors cause ICAM to return control inline following the macroinstruction issued. In no case is the macroinstruction request completed.

  - The error return address specified in NETREQ or NATTACH instruction is outside your program region.

  - The NETREQ or NATTACH table is outside your region.

- DUSTERR=INLINE Specified

  When you specify DUSTERR=INLINE and an error occurs, returns are always inline for all DUST functions. For GETCP and PUTCP message processing functions, error return procedures remain the same as when you don't specify DUSTERR=INLINE. That is, errors are returned to the ERRET= address specified in the DTFCP macroinstructions. And, if no ERRET= address is specified, control is returned inline following the macroinstruction.

  Table 2-4 shows where ICAM returns control after detecting an error and the parameters passed in registers 0 and 1. Figures 2-6 and 2-7 show how successful and error returns are made for imperative macroinstructions.

- Error Analysis

  Each of the imperative macroinstructions, in conjunction with the OS/3 supervisor, generates a parameter table that contains control information concerning the processing status of the macroinstruction. When an error condition is detected during processing, the parameter table is updated to relate the error condition, and processing of the instruction is halted. The specific error conditions that can be detected during processing are described with the macroinstructions.

  When control is returned inline or passed to an error address, register 1 points to the address of the parameter table generated by the macroinstruction. By expanding TN#DSECT, TM#DSECT, and TU#DSTZ, you can access specific fields in the various tables. The first half word of the parameter table contains the error code. (See Table 2-5.) You can also find the error code in bytes 2 and 3 of register 0.

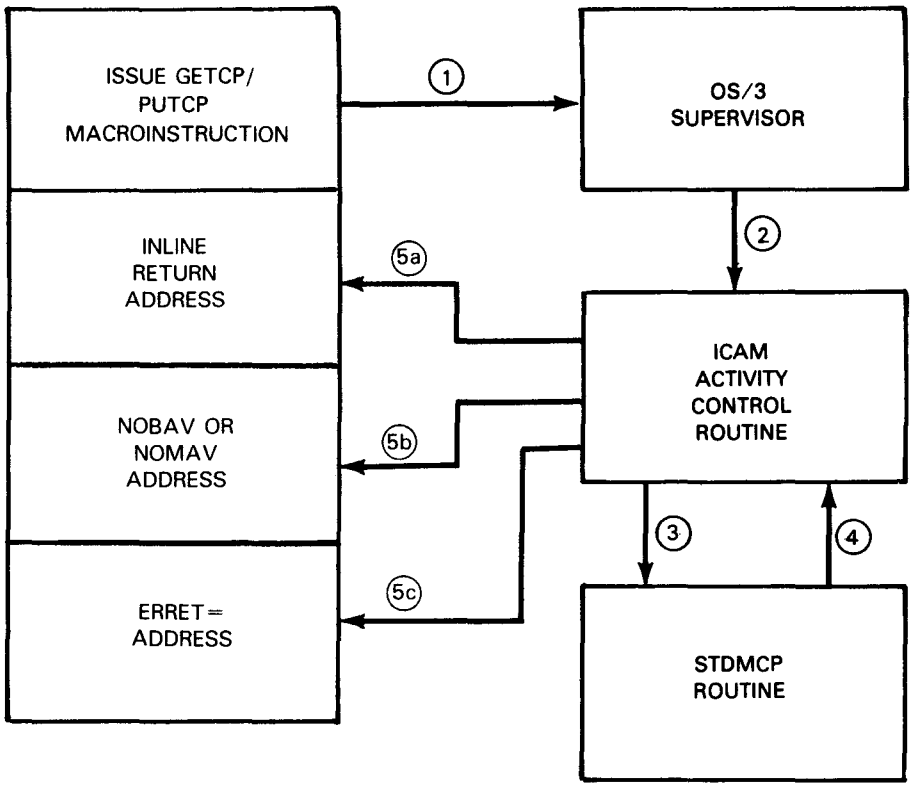
Table 2-4. Error Return Locations and Parameters Passed When Specifying or Not Specifying DUSTERR=INLINE

Macroinstructions Processed	Control Return Location		Parameters Passed	
	DUSTERR= Not Specified	DUSTERR= Specified	Register 0	Register 1
CCACPY	NETREQ/ NATTACH ERRET=	Inline	Error half word loaded from parameter table	CCACPY parameter table address
GAWAKE	Inline		Error codes	GAWAKE parameter table address
LNEREL	NETREQ ERRET=		Error half word loaded from parameter table	LNEREL parameter table address
LNEREQ				LNEREQ parameter table address
NATTACH	NATTACH ERRET=			NATTACH parameter table address
NDETACH				NDETACH parameter table address
NETREL	NETREQ ERRET=			NETREL parameter table address
NETREQ				NETREQ parameter table address
QCLEAR	NETREQ/NATTACH ERRET=			QCLEAR parameter table address
QDEPTH				QDEPTH parameter table address
QHOLD				QHOLD parameter table address
QRELSE				QRELSE parameter table address
QTRANS				QTRANS parameter table address
RELEASM				RELEASM parameter table address
SESCON	Inline		Error codes	SESCON parameter table address
TRMREL				TRMREL parameter table address
TRMREP	NETREQ/NATTACH ERRET=		Error half word loaded from parameter table	TRMREP parameter table address
GETCP	DTFCP ERRET= (if specified) or Inline (if not specified)			DTFCP table address
PUTCP				
PUTCP output delivery notification	NETREQ/NATTACH ERRET=		Status bytes	Delivery parameter
Line-down notification	NETREQ ERRET=		Error status	Line name
Unidentified terminal on line				
Control datagram or user-supplied datagram notice	GAWAKE ENTRY=		Address of datagram input buffer	

NOTES:

1. Specifying or not specifying DUSTERR=INLINE is correct only when your program is coded to match the parameter you supply.

2. For details on output delivery notification and line-down notification, see 2.9 and 2.10.



NOTES:

① Program issues GETCP or PUTCP macroinstruction.

② OS/3 supervisor passes control to ICAM's activity control routine.

③ ICAM activity control routine releases control to STDMCP to process the request.

④ STDMCP processes request and returns control to activity control routine.

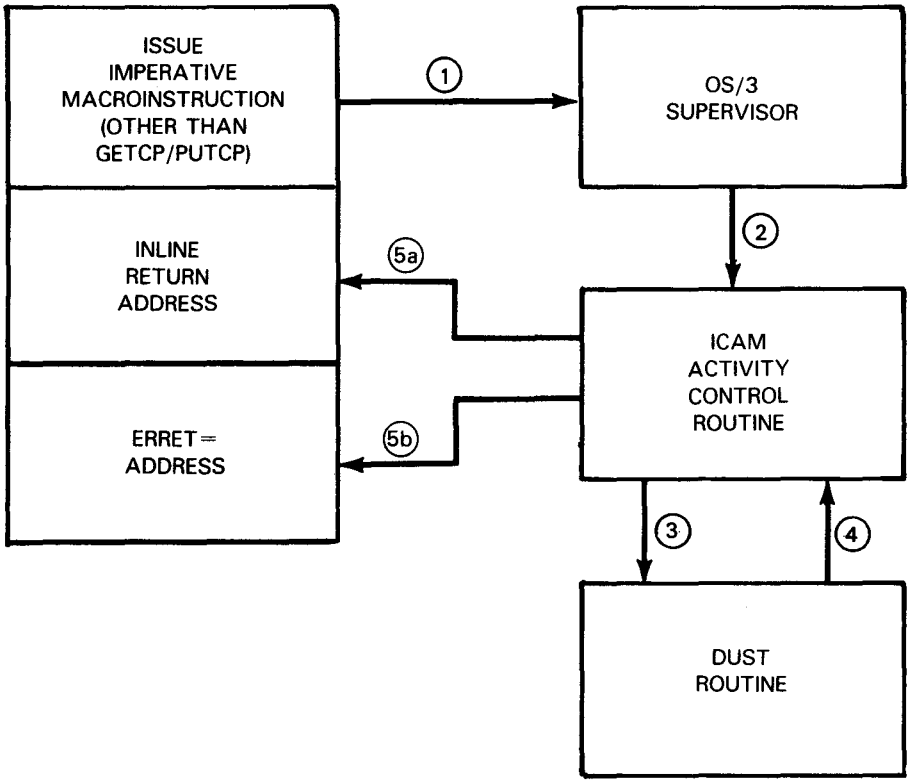
⑤ Activity control routine schedules control to program:

ⓐ Inline if successful or if an error occurs and DUSTERR=INLINE is specified

ⓑ At a specified NOBAV or NOMAV address if error occurs due to no message available or no buffer available

ⓒ At a specified ERRET= address if an error occurs and DUSTERR=INLINE is *not* specified

Figure 2-6. ICAM Message Processing Functions (GETCP/PUTCP), Successful and Error Returns



NOTES:

① Program issues imperative macroinstruction other than GETCP or PUTCP.

② OS/3 supervisor passes control to ICAM's activity control routine.

③ ICAM activity control routine releases control to DUST to process the request.

④ STDMCP processes request and returns control to activity control routine.

⑤ Activity control routine schedules control to program:

- ⓐ Inline if successful or if error occurs and DUSTERR=INLINE is specified
- ⓑ At a specified ERRET= address if an error occurs and DUSTERR=INLINE is not specified

Figure 2-7. ICAM DUST Function Processing, Successful and Error Returns

Table 2-5. TQ#x Labels for Mapping Common Part of DUST Function Tables

Label	Length	Content
TQ#xERR	OH	Error half word
TQ#xER1	XL1	Error byte 1
TQ#xER2	XL1	Error byte 2

2.13. INPUT MESSAGE ARRIVAL NOTIFICATION

The input message arrival notification feature lets you specify a list of message sources called a notification list, or *notlst,* in a GETCP request, and receive a notice when a message is queued to one of the sources you name in the notlst. The feature is supported for both dedicated and global networks.

When you issue the GETCP request, ICAM scans the queues associated with the sources you specify, and if a message is found, a notice is returned in an 8-byte work area you define in your program. The message itself is not returned; you must issue a subsequent GETCP to obtain the message. If a message is not found, the GETCP is deferred, and you are notified when a message is queued to one of the sources in the notlst.

Use the input message arrival notification GETCP in much the same way as you would any other GETCP, except that you must:

- set TM#PINPN in the TM#PDTF field of the related DTFCP;
- point to the notlst using the NOTLST operand in the DTFCP macroinstruction; or
- move the address of the notification list into TM#PNAM.

The format of the notlst is shown in Figure 2-8, and the format of the work area you need is shown in Figure 2-9. Each entry in the list must be four bytes. You must place the number of 4-byte entries in the notlst into the first 2 bytes of the work area before you issue the GETCP. When the GETCP is completed, bytes 2-5 contain the name of the source containing the message and byte 6 contains the binary value of the queue priority. Binary 1 is the lowest priority, and binary 3 is the highest.

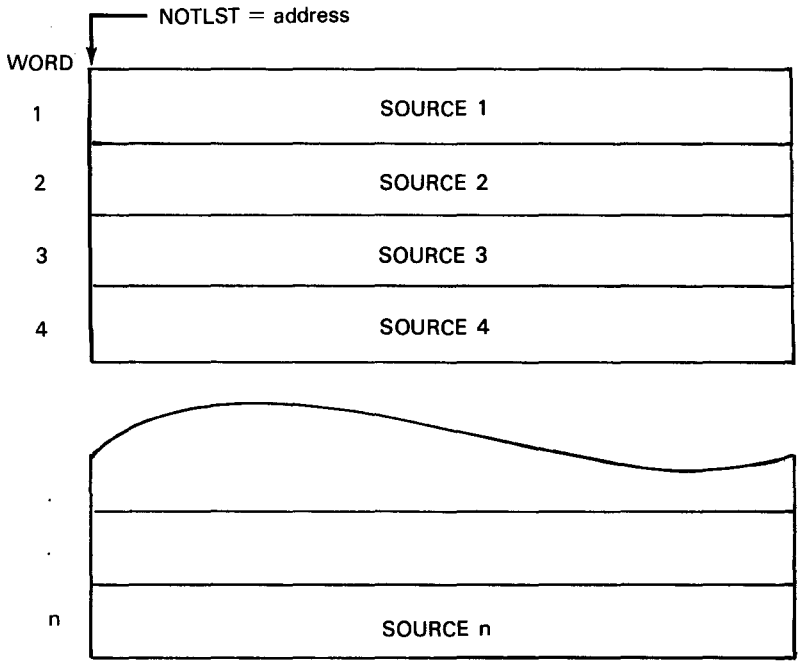


Figure 2-8. Notification List (notlst) for Input Message Arrival Notification

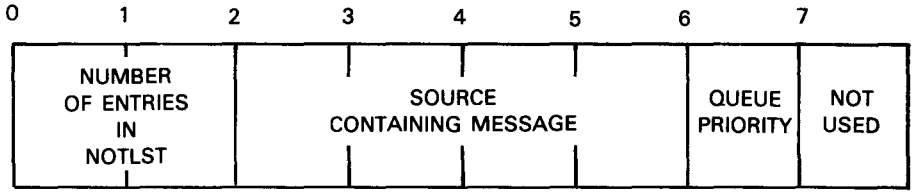


Figure 2-9. Work Area for Input Message Arrival Notification

2.14. DATE AND TIME STAMP

The date and time stamp feature provides the date and time a message is queued in special fields of the related DTFCP when you issue a GETCP request.

Select this feature by specifying the FEATURES=(DATIME) operand in the CCA macroinstruction in your network definition and the DATIME=YES operand in the process file DTFCP macroinstruction in your program.

Before you issue the GETCP, you must set TM#PDTSP in the TM#DTF field of the related DTFCP. When the GETCP is executed, ICAM copies the date and time from the message header and places them in the TM#PDATE and TM#PTIME fields of the DTFCP. Date and time are returned on only the first segment of a multisegment message.

The date and time are returned in packed format as follows:

Field	Operation	Operand	Format
TM#PDATE	DS	PL4	0YYMMDDF
TM#PTIME	DS	PL5	0HHMMSSTTF

where:

YY = year
MM = month
DD = day
HH = hours
MM = minutes
SS = seconds
TT = hundreds of seconds
F = sign

NOTES:

1. *TM#PDTSP must be maintained for subsequent GETCP requests.*

2. *TM#PDTER is set in TM#PIND+1 if the date and time are not returned with a message. This could happen if you set TM#PDTSP and forgot to specify FEATURES(DATIME) in the CCA macroinstruction.*

2.15. COMMUNICATIONS AWAKE FACILITY

The ICAM communications awake facility permits any communications program or noncommunications program to activate (awake) any registered communications program. This facility also permits your programs to optionally pass data or parameters to a registered program in the form of a message called a datagram.

Before a program can be activated and, optionally, receive a datagram, it must register with ICAM. This is done as follows:

- The GAWAKE operand in the CCA macroinstruction must be specified as YES when you generate your network definition.

- The program must issue a NETREQ or NATTACH macro.

- The program must issue a GAWAKE macroinstruction with the TYPE operand specified as INPUT. This is done to specify the address where the program is to receive control when it is activated, to indicate the address of the area that will contain the datagram when the program is activated, and the length of the datagram storage area.

When the GAWAKE macro with the TYPE operand specified as output is issued, ICAM moves the datagram into one or more network buffers associated with the program to be awakened and queues it to a facility table associated with the named receiving program. No delivery notice is returned to the sending program; thus, there is no guarantee that the datagram is delivered to the receiving program. When the receiving program becomes idle (i.e., it is in a CYIELD condition), the ICAM control program transfers the datagram into the storage area of the receiving program and activates it.

Note that the GAWAKE macroinstruction serves two purposes: to register a program, and as the vehicle to awake and optionally transfer a message to a program.

2.16. MESSAGE ERROR RECOVERY PROCEDURE

ICAM enables you to construct an error recovery procedure that recovers output messages that are undeliverable. Two methods are available; one requires the use of ICAM's MPPS and the other does not. Both require that you specify a process file that receives the undeliverable messages.

If you use MPPS, you can reroute messages to a terminal's alternate destination (not multiple destination messages unless on disk), to another terminal, or to a process file. These actions are taken based upon a group of predefined error conditions you indicate in your MPPS definition.

If you don't use MPPS, you can recover undeliverable messages by specifying the process file you defined to handle them in the ALTD operand of the TERM macroinstruction of your network definition.

In either case, your program is required to issue GETCP macroinstructions periodically to remove any messages from the process file.

2.17. HOW TO SPECIFY ICAM DSECTS

ICAM uses DSECTs to define labels and equate values it uses to process messages and to perform other functions. You use many of these same labels and values in your program, for example, when you specify a Put or Get request (e.g., a PUTCP or GETCP macroinstruction), a file definition (e.g., a DTFCP macroinstruction), or when you ask ICAM to change the environment (e.g., activate a line (LNEREQ) or clear a queue (QCLEAR)) by issuing one of the DUST macroinstructions.

In order to perform any ICAM function, the appropriate DSECT must be available in your program. You include them by assembling the procs TM#DSECT, TN#DSECT, and TU#DSTZ.

- TM#DSECT calls the DSECTs your program uses to interface with ICAM, such as GETCP, PUTCP, DTFCP, etc. If you don't call TM#DSECT, ICAM will include it for you.

- TN#DSECT calls the DSECTs that ICAM uses in its own processing (you always require at least some of the DSECTs in TN#DSECT). TN#DSECT also calls the DSECTs needed to process the NETREQ, NETREL, LNEREQ, and LNEREL DUST macroinstructions in your program.

- TU#DSTZ calls the DSECTs needed to process the CCACPY, QCLEAR, QDEPTH, QHOLD, QRELSE, QTRANS, RELEASM, and TRMREP DUST functions. If you use any of the DUST functions, you must assemble TU#DSTZ.

When you assemble one of these procs, and you don't specify any operands, all DSECTs and equates available from that proc are assembled into your program. Therefore, you can save assembly time and lots of paper by specifying only the operands that relate to the functions you need. These are listed in Tables 2-6 and 2-7. When you do this, only the DSECTs you request are assembled in your program.

The MUST DSECTs required to process GETCPs and PUTCPs (see TM#DSECT GETPUT in Table 2-6) are automatically called when you assemble your first GETCP or PUTCP macroinstruction. You don't need to call them explicitly. However, you may want to assemble all of the ICAM DSECTs your program needs at the beginning of your program for easy reference. If so, call each one you need with the proc calls. An ICAM DSECT is never assembled more than once into your program; however, you should avoid calling the same DSECT more than once.

An example follows the tables that list the proc call definitions and shows how to assemble some of the ICAM DSECTs and obtain listings of them for reference.

The formats for calling the procs are:

LABEL	△OPERATION△	OPERAND
	TM#DSECT	See Table 2-6

LABEL	△OPERATION△	OPERAND
	TN#DSECT	See Table 2-7

LABEL	△OPERATION△	OPERAND
	TU#DSTZ	(No operands) See Table 2-8

Table 2-6. TM#DSECT Proc Call Details

DSECT Name	Description	User Interface	Operand Specification*	
			Individual Selection	Group Selection
TM#PRCS	Process File DTF	STDMCP	PRCS	GETPUT
TM#DEST	Destination File DTF	STDMCP	DEST	GETPUT

*Executing the TM#DSECT call with no operands defaults to *all* requested.

Table 2-7. TN#DSECT Proc Call Details

DSECT Name	Description	User Interface	Operand Selection	
			Individual Selection	Group Selection
TN#ARP	Activity Control SVC Decode ARP	N/A	ACTARP	BACTGRP
TM#ARP	MUST ARP	N/A	ARP97	*
TN#ACTB	Basic Activity Control Table	N/A	BASTAB	BACTGRP
TN#BPOOL	ARP/Buffer Pool Control Table	N/A	BPOOL	*
TN#CNTRL	CCA Control Section	N/A	CCACON	CCAGRP
TN#DCT	Auxiliary Device Control Table	DUST	DCT	CCAGRP
TN#EDTBL	Destination Table	N/A	DESTBL	CCAGRP
TN#DLIST	Distribution List	N/A	DLIST	CCAGRP
TN#GEN	ICAM General Information Table	N/A	GENTAB	BACTGRP
TN#GTCBS	User TCB Directory and Activity Queue Table	N/A	GENTAB	BACTGRP
TN#LCT	LINE Macro Table (LCT)	N/A	LCT	CCAGRP
TN#MSG	Network Buffer Prefix	N/A	MSGPRE	*
TN#VARP	Overlay Control/Operator Communications ARP	N/A	OVARP	*
TN#FPRCS	PRCS Macro Table	N/A	PRCS	CCAGRP
TN#TCT	TERM Macro Table	N/A	TCT	CCAGRP
TN#ARTME	ARP Timer	N/A	TIMARP	DDIGRP
TN#QCT	Queue Control Table	N/A	QCT	CCAGRP
TN#KGAWI	GAWAKE Input Parameter List	STD	**	GAWAKE
TN#KGAWO	GAWAKE Output Parameter List	STD	**	GAWAKE
Error code equates for NETREQ/NETREL, and LNEREQ/NETREL DUST macros				
TQ#DSCTS	General DUST Macro Table			
TQ#NET	NETREQ/NETREL	DUST	**	DUST
TQ#LINE	LNEREQ/LNEREL	DUST	**	DUST

* May only be obtained by individual selection.
** May only be obtained by group selection.

NOTE:

If you call TN#DSECT and do not specify any operands, all DSECTs are assembled.

Table 2-8. TU#DSTZ DSECT Names

DSECT Name	Description
TQ#QDSCT	Parameter List for QHOLD, QRELSE, QDEPTH, QCLEAR
TQ#QDWA	QDEPTH Output Definition
TC#CCINP	CCACPY Terminal Input Definition
TC#CCOTP	CCACPY Output Definition

Example:

This is an example of a job stream to assemble a DSECT.

```
1         10    16                                                  72
// JOB JHB
// DVC 2Ø
// LFD PRINT
// ASM
/$
 START
 TM#DSECT GETPUT              GET PROCESS FILE DTF (TM#PRCS) AND
*                             GET DESTINATION FILE DTF (TM#DEST)
 TN#DSECT DUST                GET DUST MACROS AND EQUATES
 END
/*
/&
// FIN
```

2.18. USER ISLAND CODE CONSIDERATIONS

All user SVC calls to ICAM from island code must be done with IRL requested. If the user fails to specify an IRL, or if the SVC is a CYIELD macro call, the user is permanently suspended.

2.19. HOW TO FORMAT OUTPUT MESSAGES IN YOUR PROGRAM

When you need to send a message to a terminal, you need to prepare it in your program so that it is displayed or printed correctly on the receiving terminal. This is a task shared by ICAM's remote device handlers and your program.

The ICAM remote device handlers provide:

- Text delimiting (framing) characters

  Text delimiting characters include start of header (SOH), start of text (STX), end of text (ETX), and others. You never need to specify these characters in your program.

- All necessary control characters when you use DICE sequences

  Control characters control the movement of the cursor on a video display terminal or the position of the print carriage on a hard-copy device.

- Code translation

When your program releases a message to ICAM for transmission, the EBCDIC characters are translated into the code used on the communications line to which the receiving terminal is attached. You have the option of specifying your own translation (or no translation) table in your network definition. See the ICAM network definition and operations user guide, UP-8947 (current version) to see how to do this.

- Time fill

  After a line is printed, certain terminals such as teletypewriters require time to move their carriages to the beginning of a new line before resuming printing. ICAM provides this time by inserting the appropriate time fill characters in the message.

You format output messages in your program in three ways:

1. By providing all control characters. If you do this, ICAM still provides time fills for you when necessary.

2. By using DICE sequences in your message. If you do this, ICAM automatically converts the DICE sequences into the appropriate control characters for the receiving device. DICE is helpful when you need to send a message to several different kinds of terminals, for example, to a UTS 4000, and to a teletypewriter.

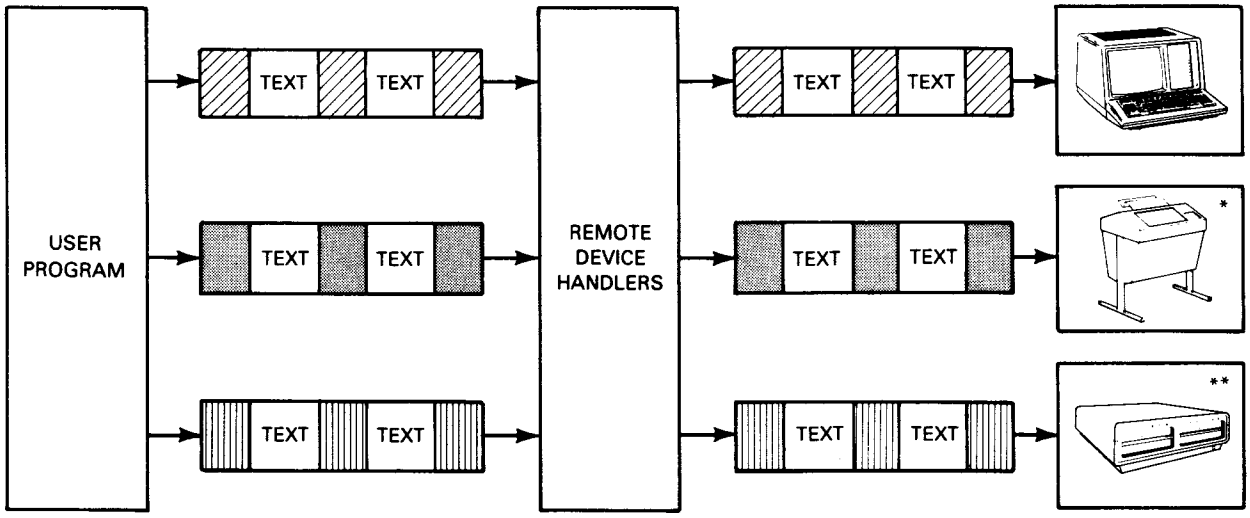
3. By a combination of both.

2.19.1. Using DICE to Format Messages

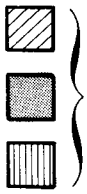
For output, your program can use either of two methods to control the format of a message displayed at a terminal.

1. By embedding format control characters, the message text is directed to each specific terminal. Obviously, if you do this, your program must include a different formatting routine for each type of terminal; this is illustrated in the following diagram:

OUTPUT TEXT AND CONTROL CHARACTERS



LEGEND:



Terminal-oriented control characters

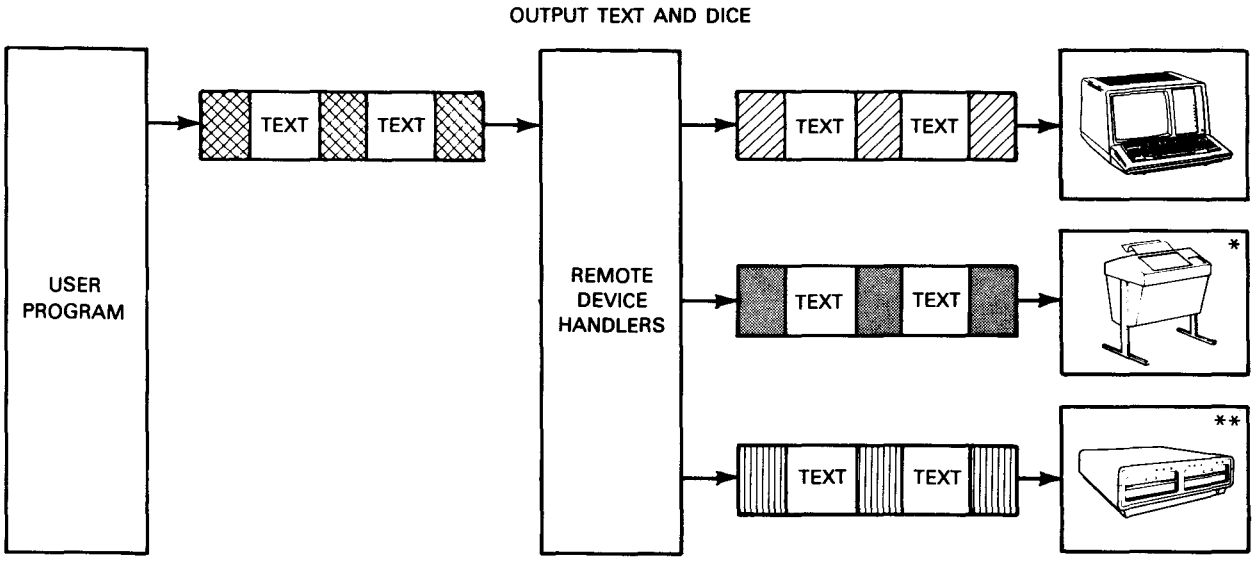
* COP attached to a UTS 400

** Diskette attached to a UTS 400 with screen bypass

2. By embedding DICE sequences, the control of format for various types of terminals and auxiliary devices is simplified. The remote device handler (RDH) converts DICE sequences to control characters for each destination terminal. Some of the control character functions are:

   - line feed – cursor movement to the first space of a new line;
   - form feed – cursor to the home position of a new page;
   - carriage return – cursor to the beginning of the same line; or
   - cursor movement to a specific row and column on a display.

   You can place DICE sequences anywhere in a message to accomplish the control you want. As you can see by the following illustration, formatting is easier when you use DICE.



LEGEND:

DICE characters

Terminal-oriented control characters

* COP attached to a UTS 400

** Diskette attached to a UTS 400 with screen bypass

For input, control characters received in a message are converted into DICE sequences by the RDH. For certain terminals, your program can analyze these DICE sequences to determine cursor position. In addition, input DICE is handy for message switch application because control characters in each input message are converted to DICE sequences. The RDH converts these sequences into the appropriate control characters for the destination terminal.

You can turn DICE on or off at network definition time with the DICE operand on the TERM macroinstruction.

```
DICE=[{ON }]
      {OFF}
```

The default is DICE=(ON).

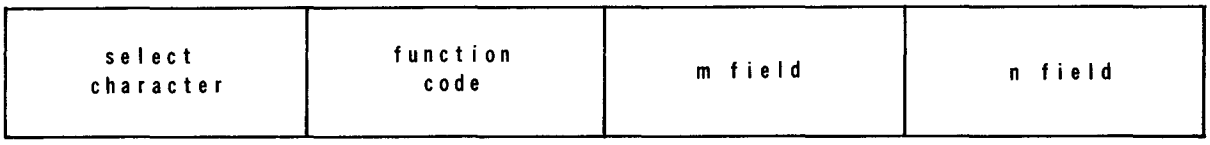
1. DICE=(ON) tells the RDH to create input DICE according to your input terminal's cursor movements.

2. DICE=(OFF) tells the RDH not to create input DICE. In this case, your program receives format control characters in the text message. The characters your program receives then depend on the *kind* of terminal that sent the message.

2.19.2. Format of DICE Sequences

The 4-byte format of a DICE sequence is as follows:

Format:



where:

`select character`
Is a hexadecimal character (10_{16}) designating the start of a DICE sequence. This character, a data link escape (DLE) control character in EBCDIC, must be used only to designate the start of a DICE sequence.

`function code`
Defines the device control sequence that is recognized by the RDHs on input. On output, this code is a 1-byte field defining the operation to be performed on the text message. DICE function codes are listed in Table 2-9.

`m field and n field`
These fields are treated as parameters to the DICE function code; their actual definition varies and is determined by the individual DICE macroinstruction. Generally, m relates to vertical positioning and n applies to horizontal positioning.

2.19.3. DICE Macroinstructions

DICE macroinstructions let you create DICE sequences (DICE constants) in the same way you would create constants in your program. That is, when the assembler expands a DICE macroinstruction, it creates a constant at that location. On output (when your program is ready to send a message), your program moves the DICE constants created from the DICE macroinstructions into the appropriate places in your message before it issues the output request. The RDH converts the DICE constants into the corresponding control characters to produce the necessary positioning.

On input, DICE sequences are automatically created by the RDHs unless you specify DICEOFF in your network definition. Table 2-9 lists the DICE macroinstructions, function code generated, and m and n coordinates as they apply to particular devices on input and output.

You must specify m and n coordinates in your program according to the absolute and relative values expressed in Table 2-9. m_a and n_a are absolute values of m and n. m_r and n_r are relative displacements of m and n. For CRT terminals, the home position is $(m_a,n_a)(1,1)$. For character- or page-oriented devices that allow position to top of form, the top-of-form position is $(m_a,n_a)=(1,1)$.

- Absolute Positions

  Absolute positions of m and n may range as follows:

  m_a ranges 1 to r

  where:

  r = maximum number of rows (CRT) or maximum number of lines per page.

  n_a ranges 1 to c

  where:

  c = maximum number of columns (CRT) or maximum number of character positions per line.

- Relative Displacements

  Relative displacements of m and n may begin at zero and range to the bottom and right margin of the screen or page.

- If a value of m or n falls outside of the legal range, that value of m or n will cause the following action:

  m or n = 0 is interpreted as m or n = 1

  Specifying an absolute or relative value for m or n that is greater than the screen or page size causes unpredictable results.

2.19.4. DICE Code Generation

Macroinstructions are provided to generate the DICE codes.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	dice-macroinstruction	m,n

Label:

`[symbol]`
An optional alphanumeric character string, from one to eight characters long, that identifies the specific instruction line.

Operation:

`dice-macroinstruction`
You specify the appropriate name from the macroinstruction column of Table 2-9 for the desired DICE sequence.

Operand:

Positional Parameter 1:

`m`
A decimal number (0 to 255) indicating the number of lines or rows the terminal should advance before starting output of the message (Table 2-9).

Positional Parameter 2:

`n`
A decimal number (0 to 255) indicating the number of spaces or columns to the right the terminal should space before starting output of the message (Table 2-9).

Examples:

```
1. NEWLINE   ZO#POS Ø,Ø
2. COORDI    ZO#COORD 5,1Ø
```

1. This DICE sequence causes movement to a new line.
2. New text starts at line 5, column 10 due to this DICE.

Table 2-9. DICE Input/Output Commands, Codes and Device Interpretation (Part 1 of 5)

DICE Macro-instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices (1)	CRT Devices	Page Printing Devices (n is not interpreted)	Communications Output Printer (COP) (1)
ZO#COORD	Set coordinates	01_{16}	INPUT	m	n	Not used	m and n represent the start-of-entry (SOE) cursor coordinates.	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. (2)	Move cursor to row m and column n. (5)	Action is optional. (2)	Action is optional. (2)
ZO#FORM	Forms control	02_{16}	INPUT	01	01	Form feed	Form feed	Not used	Not used
			OUTPUT	m_a	n_a	Form feed, carriage return, and advance to line m and column n (m—1 line feeds and n—1 spaces to the right)	Move cursor to row m and column n. (5)	Top of form and advance to line m (m—1 line feeds)	Form feed, line feed, and advance to line m and column n (m—1 line feeds and n—1 spaces to the right)
ZO#FORMC	Forms control with clear; unprotected data	03_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. (2)	Move cursor to row m and column n, and clear unprotected data to end of screen. (5)	Action is optional (2)	Action is optional. (2)
ZO#POS	New line control	04_{16}	INPUT	00	00	Carriage return, line feed (6)	Cursor return (6)	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return, line feed, followed by m line feeds and n spaces to the right.	Move cursor to beginning of next line. Then move cursor m lines (6) down and n columns to the right.	Advance (m+1) lines.	Line feed, followed by m line feeds and n spaces to the right.

Table 2-9. DICE Input/Output Commands, Codes and Device Interpretation (Part 2 of 5)

DICE Macro-instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices (1)	CRT Devices	Page Printing Devices (n is not interpreted)	Communications Output Printer (COP) (1)
ZO#POSC	New line control with clear	05_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return, line feed, followed by m line feeds and n spaces to the right	Same as 04_{16} except area between start and end positions is cleared. (6)	Advance (m+1) lines.	Line feed, followed by m line feeds and n spaces to the right.
ZO#CUR	Current position control	06_{16}	INPUT	01	00	Line feed (6)	Not used	End of input card	Not used
			OUTPUT	m_r	n_r	m line feeds and n spaces to the right	Move cursor m lines down and n columns to the right. (6)	Advance m lines.	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (4)
ZO#CURC	Current position control with clear	07_{16}	INPUT	—	—	Not used	Line feed	Not used	Not used
			OUTPUT	m_r	n_r	m line feeds and n spaces to the right	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (3) (6)	Advance m lines.	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (3)
ZO#BEG	Beginning of current line control	08_{16}	INPUT	00	00	Carriage return (6)	Not used	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return followed by m line feeds and n spaces to the right	Move cursor to (6) beginning of current line. Then move cursor m lines down and n columns to the right.	Advance m lines.	m line feeds and n spaces to the right.

Table 2-9. DICE Input/Output Commands, Codes and Device Interpretation (Part 3 of 5)

DICE Macro-instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices (1)	CRT Devices	Page Printing Devices (n is not interpreted)	Communications Output Printer (COP) (1)
ZO#TABS	Set tab stop at an absolute position (4)	09_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	No line feed, space to right.	Set tab stop at row m and column n. (5)	Advance m lines.	Not used
ZO#FORMA	Forms control with clear; protected/ unprotected data	$0A_{16}$	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. (2) (6)	Move cursor to row m and column n and clear protected/unprotected data to end of screen. (6)	Action is optional. (2)	Action is optional. (2)
ZO#ERSLN	Erase to end of line	$0B_{16}$	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	No action (6)	Cursor does not move. Unprotected data to the end of a line or to the end of the first unprotected field is cleared, whichever comes first. (6)	Advance 0 lines.	Not used

NOTES:

(1) Most character-oriented terminals can be strapped to handle the carriage return (CR) character and the line feed (LF) character as follows:

- CR
  1. print mechanism moves to beginning of the same line; or
  2. print mechanism moves to the beginning of the same line followed by a line feed.
- LF
  1. line feed (no column change); or
  2. line feed followed by return of the print mechanism to the beginning of the new line.

Table 2-9. DICE Input/Output Commands, Codes and Device Interpretation (Part 4 of 5)

To achieve device independence between terminal types, the character-oriented terminals must use the first option for CR and the first option for LF if the device macroinstruction is ZO#CUR or ZO#BEG.

The first option should be used if the character-oriented terminals are a part of a message switch environment.

Certain terminals do not have a form feed capability (i.e., some TTY terminals). For these terminals, the DICE expressions that specify form feed will result in line feed instead.

② The set coordinates macroinstruction (ZO#COORD) or the forms control with clear macroinstruction (ZO#FORMC), when acted upon by character-oriented or page-printing terminals, will vary in its actions, depending on the use of the DICE keyword parameter of the TERM macroinstruction at network definition time:

```
TERM  ...,DICE=[{FORMS   }],...
               {NEWLINE}
```

If FORMS is specified, the set coordinates macroinstruction will be interpreted as the forms control macroinstruction.

If NEWLINE is specified, the set coordinates macroinstruction and the forms control with clear macroinstruction will result in a carriage return, line feed for character-oriented terminals, or advance one line for page-oriented terminals (m and n are not interpreted).

If the DICE parameter is not specified, the default option is NEWLINE.

③ The UNISCOPE display terminal suppresses nonsignificant spaces on each line (except for the line containing the cursor) when text is transmitted to the processor or printed locally on the COP or TP.

Your program may send data to the UNISCOPE screen containing significant blank segments that include the last column of the screen. If this data is transmitted from the terminal to the processor or is printed locally on the COP or TP, the blank segments must consist of nonspace characters that are nondisplayable. The DC3 character meets these qualifications. The ICAM interface provides your program with the capability to prevent nonsignificant space suppression on the UNISCOPE display terminal. The "current position control with clear" is the only DICE macroinstruction that can be used to perform a clear function if your program is preventing nonsignificant space suppression.

NOTE:

The ASCII-to-EBCDIC translation table is modified so that the DC3 character is translated to space 40_{16} for input from the UNISCOPE display terminal.

④ When using DICE function code 09_{16} for setting a tab stop, m=0 and n=0 will result in a tab stop being placed at the current cursor location (no cursor positioning is performed). This applies to UNISCOPE and UTS 400 devices only. For TTYs and DCT 500 terminals, a space character is inserted.

If m or n is greater than the maximum allowable m or n, action will vary depending on the remote terminal:

- UNISCOPE display terminals - wraparound will occur on screen.
- Character-oriented terminals - will give different results depending on the characteristics of the device.

⑤ For an IBM 3270 display terminal, the m and n values are used to set buffer address to [80 x (m-1) + (n-1)] or [40 x (m-1) + (n-1)].

⑥ DICE functions not supported for the IBM 3270 display or printer.

⑦ The following applies to the use of DICE with UNISCOPE, UTS 400, and UTS 4000 terminals.

Table 2-9. DICE Input/Output Commands, Codes and Device Interpretation (Part 5 of 5)

- If you specify DICE=OFF

  In a message containing a start-of-entry character (RS), the following data is received in the text portion of your input work area:

E	V	Y	X	N	S	R		} ASCII
S	T			U	I	S	rest-of-text	
C				L				} hexadecimal (EBCDIC)
27	0B	yy	xx	00	0F	1E		

- If you specify DICE=ON

  Control character sequences are converted to DICE sequences. For example, in a message containing a start-of-entry character (RS), the following text is received in the text portion of your work area:

4-character-dice-sequence	R S	rest-of text

Note that start-of-text (STX) and end-of-text characters (ETX) are always removed by the remote device handlers; they are never supplied to your program as text.

2.19.5. Interpretation of DICE

When using DICE, your program does not need to be aware of the terminal type. A particular DICE denotes the same positioning on any terminal. There are some exceptions that result from limitations of the terminal.

The interpretation of a DICE by the RDH is controlled by the following factors:

1. DICE function code
2. DICE m and n fields
3. The terminal involved
4. The particular device on the terminal being used

The ICAM RDHs currently provide device-independent support for three classes of remote terminal devices.

1. Hard copy character-oriented devices, such as the SPERRY UNIVAC Data Communications Terminal 475 (DCT 475), Data Communications Terminal 500 (DCT 500), Data Communications Terminal 524 (DCT 524), Data Communications Terminal 1000 (DCT 1000), and Universal Terminal System 10 (UTS 10); TELETYPE* teletypewriter models 28, 32, 33, 35, 37.
2. Hard copy page printer type devices, such as the SPERRY UNIVAC 1004 Card Processor System, Data Communications Terminal 2000 (DCT 2000), and 9200/9300 Systems, and the IBM 2780.

**Registered trademark of Teletype Corporation*

3. CRT-type terminals, such as the UNISCOPE 100 and 200, the SPERRY UNIVAC UTS 400/UTS 4000 terminals, and the IBM 3277 terminal.

Table 2-10 defines the primary output device and the primary input device for each terminal type.

Table 2-10. DICE Primary Devices

Terminal Type	Primary Output Device	Primary Input Device
Character-oriented terminals	Printer	Keyboard
Page printing terminals	Printer	Card reader
CRT terminals	Screen	Keyboard

In addition to the specified primary devices, each terminal has the ability to support one or more auxiliary devices. The auxiliary devices suggested by each terminal are listed in Table 2-11.

Table 2-11. DICE Usage for Auxiliary Devices (Part 1 of 2)

Remote Terminals	Auxiliary Device	DICE Usage
UNISCOPE/ UTS 400/ UTS 4000	See Section 3.	DICE is applied to the COP. ①
DCT 1000	Card reader/card punch Paper tape reader/punch	DICE is applied as if the output/input is to/from the primary device, even though it is for the auxiliary device. ②
DCT 500/TTY	Paper tape reader/punch	
DCT 524	Tape cassette (TCS) in paper tape read and write only	
Batch terminals	Punch	DICE is used for end of network buffer sentinel. No form control action is taken.

Table 2-11. DICE Usage for Auxiliary Devices (Part 2 of 2)

NOTES:

① If the print transparent option is not used, DICE is applied to the UNISCOPE screen even though the output is sent to an auxiliary device of the UNISCOPE terminal. In this case, the format of the data printed on the COP or TP is identical to the screen format. Nonsignificant space suppression by the UNISCOPE terminal may have to be prevented to keep the formats identical.

The full capability of DICE cannot be applied to the COP because of hardware characteristics. All data to a UNISCOPE auxiliary device passes through the UNISCOPE terminal. When DICE is applied to the COP, the use of print transparent mode means that no carriage returns are transferred to the COP. Line feeds and form feeds take a storage position in the UNISCOPE storage and are nondisplayable. These characters are passed to the COP where:

- an LF causes a line feed followed by return of the print mechanism to the beginning of the new line; and
- an FF causes a page eject and positioning of the print mechanism at the beginning of the first line of the form.

The COP has no tabbing capability.

These characteristics are reflected in the interpretation of DICE output function codes for the COP as shown in Table 2-9.

For messages sent to a UNISCOPE auxiliary device with transparent transfer, the cursor to home (ESC e) sequence is inserted at the beginning of the text by the RDH.

② The control characters that are generated from the DICE macroinstructions are always created for the primary device of a character-oriented device, even though your program is sending to an auxiliary device. The message and these control characters (carriage returns, line feeds, form feeds, and spaces) will be punched/written by the output auxiliary device that was specified by your program or was switch-selected by the terminal operator. If the punched/written data is later read by the terminal's input auxiliary device, the carriage returns, line feeds, and form feeds are converted to input DICE as specified in Table 2-9.

2.19.6. Sample DICE Programs

The following two programs show you how to format DICE sequences. Figure 2-10 shows a screen matrix to let you determine placement of your message before writing your DICE sequences. Figure 2-11 shows a display terminal screen with the same message positioned on it. Figures 2-12 and 2-13 show an assembly language program with the two methods of specifying DICE sequences (hexadecimal notation and DICE macroinstructions) used to format the message in Figure 2-11.

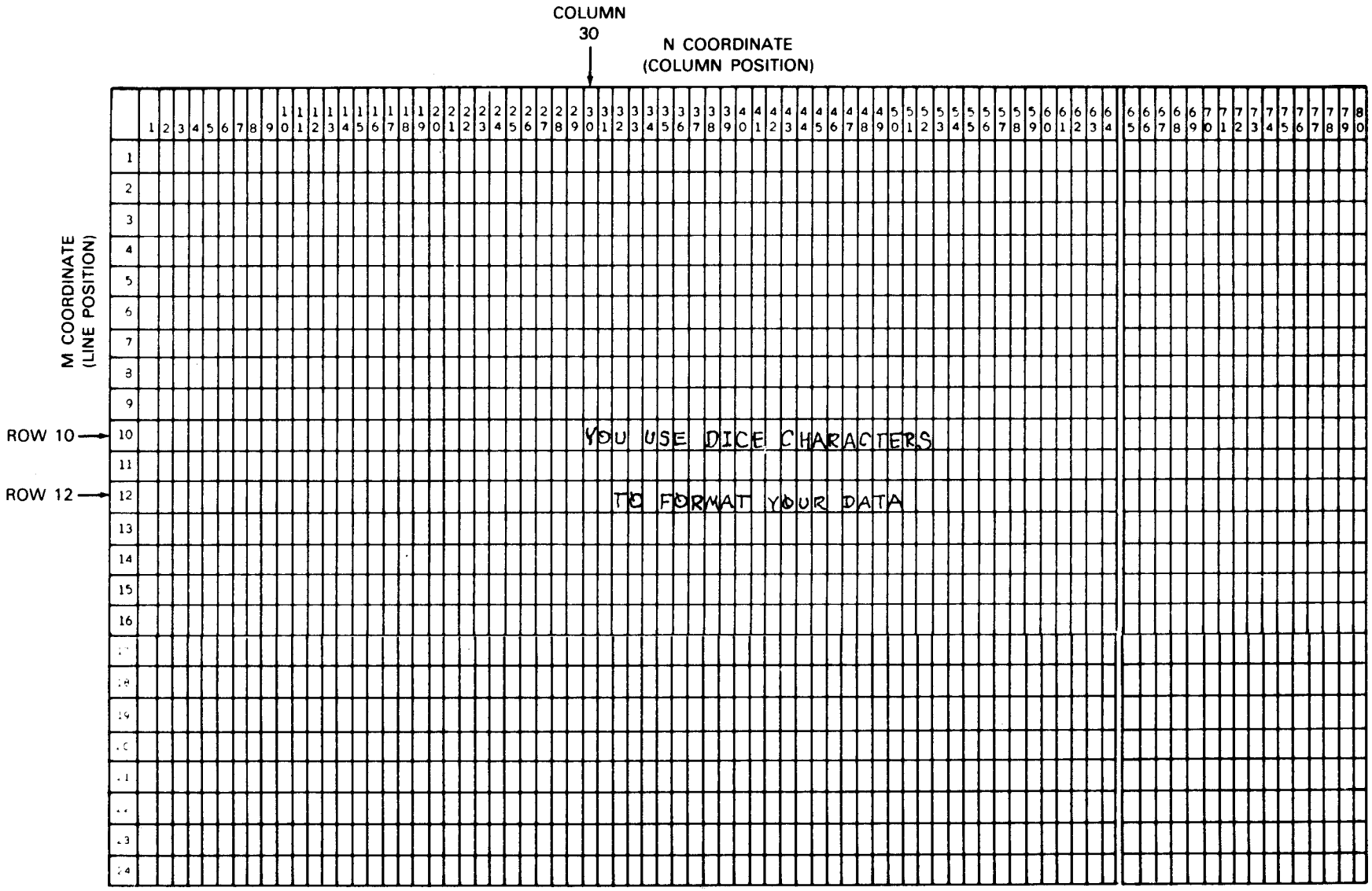


Figure 2-10. Positioning Your Message

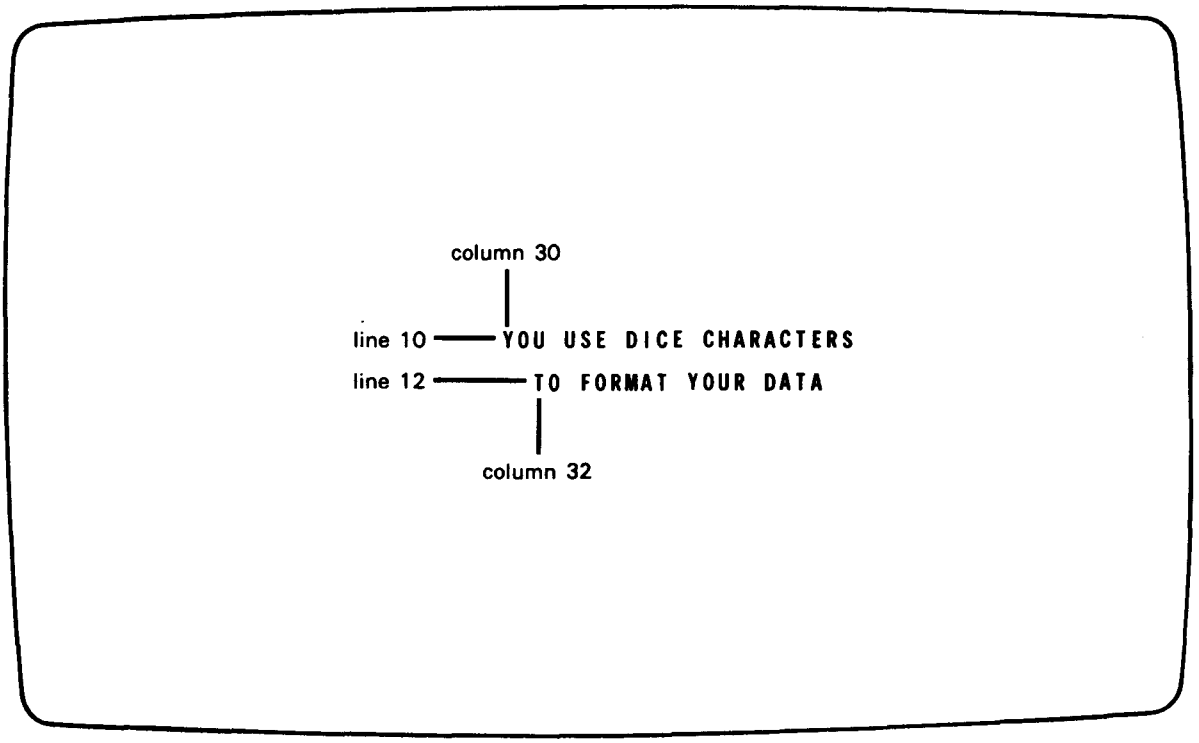


Figure 2-11. UNISCOPE Display of DICE Formatted Message

The DICE sequences used to format this message are shown in the program in Figure 2-12.

```
 1.0000 OS3HEX   START 0
 2.0000          TM#DSECT DUST
 3.0000          TM#DSECT PRCS
 4.0000          TM#DSECT DEST
 5.0000 A1       BALR  10,0
 6.0000          USING *,10
 7.0000          USING TM#PRCS
 8.0000          USING TM#DEST,3
 9.0000          USING TQ#DSCTS,4
10.0000 A2       LA    2,PRFA
11.0000 A3       XC    TM#PERR,TM#PERR
12.0000 A4       LA    3,OTPT
13.0000 A5       XC    TM#DERR,TM#DERR
14.0000 BEGIN    NETREQ NPR,ERRET=NETERR
15.0000 LOOP     OI    TM#DSEG,TM#DHOR++TM#DTND
16.0000          MVC   BUFFER(2),TXTCHAR
17.0000          MVC   BUFFER+2(2),CLEAR
18.0000          MVC   BUFFER+6(4),COORD1
19.0000          MVC   BUFFER+10(23),MSG1

20.0000          MVC   BUFFER+33(4),COORD2
21.0000          MVC   BUFFER+37(19),MSG2
22.0000 PUTIT    PUTCP OTPT,BUFFER
23.0000          XC    TM#PERR,TM#PERR
24.0000          MVI   TM#PIND,TM#PIRL
25.0000          MVC   TM#PCMPA,=A(BINGO)
26.0000          MSGIN,TXTCHAR
27.0000 GETIT    GETCP PRFA,MSGIN
28.0000          CYIELD
29.0000 BINGO    CLC   TXTIN(4),=C'DONE'
30.0000          BE    ENDJOB
31.0000          CLC   TXTIN(4),=X'84969585'
32.0000          BE    ENDJOB
33.0000          MVI   TXTIN,0
34.0000          MVI   TXTIN+1(255),TXTIN
35.0000          B     LOOP
36.0000 ENDJOB   LNEREL LNE1
37.0000 J        NETREL NPR
38.0000 K        B     DONEJOB

39.0000 NETERR   LA    1,MSG1A
40.0000          OPR   MSG1A,26
41.0000 L        SNAP  BEGIN,ENDBUFF
42.0000          B     DONEJOB
43.0000 NOGET    LA    1,MSG2A
44.0000          OPR   MSG2A,12
45.0000 M        SNAP  BEGIN,ENDBUFF
46.0000          B     DONEJOB
47.0000 NOPUT    LA    1,MSG3A
48.0000          OPR   MSG3A,12
49.0000 NOGET    SNAP  BEGIN,ENDBUFF
50.0000          B     ENDJOB
51.0000 NOBUFF   LA    1,MSG4A
52.0000          OPR   MSG4A,19
53.0000 O        SNAP  BEGIN,ENDBUFF
54.0000 DONEJOB  EOJ
55.0000          DS    OF
56.0000 PRFA     DTFCP TYPE=GT,ERRET=NOGET
57.0000          DS    OF

58.0000 OTPT     DTFCP TYPE=PT,ERRET=NOPUT,DEST=(T,TRM1),NOBAV=NOBUFF
59.0000 MSG1A    DC    CL26'ERROR ISSUING NETREQ MACRO'.
60.0000 MSG2A    DC    CL12'ERROR ON GET'
61.0000 MSG3A    DC    CL12'ERROR ON PUT'
```

Figure 2-12. Hexadecimal DICE Coding (Part 1 of 2)

```
62.0000 MSG4A    DC    CL19'NO BUFFER AVAILABLE'
63.0000 CLEAR    DC    X'10030000'
64.0000 COORD1   DC    X'10010A1E'
65.0000 COORD2   DC    X'10010C20'
66.0000 MSG1     DC    C'YOU USE DICE CHARACTERS'
67.0000 MSG2     DC    C'TO FORMAT YOUR DATA'
68.0000 TXTCHAR  DC    X'0107'
69.0000          DS    0H
70.0000 BUFFER   DC    CL256' '
71.0000 ENDBUFF  DS    0H
72.0000 MSGIN    DC    X'0000'
73.0000          DC    CL5' '
74.0000 TXTIN    DC    CL256' '
75.0000 FINISH   DS    0H
76.0000          END
```

Line	Explanation
16	Sets message length to 263
17	Moves the DICE code with the tag CLEAR into BUFFER
18	Moves the coordinates DICE code for your first message into BUFFER
19	Moves your first message line into BUFFER
20	Moves the DICE coordinates for your second message into BUFFER
21	Moves your second message line into BUFFER
22	Takes the contents of your file BUFFER and places it in an output file (line 5) destined for terminal 1
	DICE sequences specified in hexadecimal:
63	Clears screen
64	Places cursor at line 10, column 30
65	Places cursor at line 12, column 32
66, 67	Message text

NOTE:

You can specify an ERRET= address, as shown on line 14, only when you don't specify DUSTERR=INLINE in the CCA macroinstruction.

Figure 2-12. Hexadecimal DICE Coding (Part 2 of 2)

If you use DICE macroinstructions, your program appears as shown in Figure 2-13.

```
 1.0000 OS3DICE   START 0
 2.0000           TN#DSECT DUST
 3.0000           TM#DSECT PRCS
 4.0000           TM#DSECT DEST
 5.0000 A1        BALR  10,0
 6.0000           USING *,10
 7.0000           USING TM#PRCS
 8.0000           USING TM#DEST,3
 9.0000           USING TQ#DSCTS,4
10.0000 A2        LA    2,PRFA
11.0000 A3        XC    TM#PERR,TM#PERR
12.0000 A4        LA    3,OTPT
13.0000 A5        XC    TM#DERR,TM#DERR
14.0000 BEGIN     NETREQ NPR,ERRET=NETERR
```

Figure 2-13. DICE Macroinstruction Coding (Part 1 of 3)

```
15.0000 LOOP     OI   TM#DSEG,TM#DHDR++TM#DTND
16.0000          MVC  BUFFER(2),TXTCHAR
17.0000          MVC  BUFFER+2(2),CLEAR
18.0000          MVC  BUFFER+6(4),COORD1
19.0000          MVC  BUFFER+10(23),MSG1

20.0000          MVC  BUFFER+33(4),COORD2
21.0000          MVC  BUFFER+37(19),MSG2
22.0000 PUTIT    PUTCP OTPT,BUFFER
23.0000          XC   TM#PERR,TM#PERR
24.0000          MVI  TM#PIND,TM#PIRL
25.0000          MVC  TM#PCMPA,=A(BINGO)
26.0000          MSGIN,TXTCHAR
27.0000 GETIT    GETCP PRFA,MSGIN
28.0000          CYIELD
29.0000 BINGO    CLC  TXTIN(4),=C'DONE'
30.0000          BE   ENDJOB
31.0000          CLC  TXTIN(4),=X'84969585'
32.0000          BE   ENDJOB
33.0000          MVI  TXTIN,0
34.0000          MVI  TXTIN+1(255),TXTIN
35.0000          B    LOOP
36.0000 ENDJOB   LNEREL LNE1
37.0000 J        NETREL NPR
38.0000 K        B    DONEJOB

39.0000 NETERR   LA   1,MSG1A
40.0000          OPR  MSG1A,26
41.0000 L        SNAP BEGIN,ENDBUFF
42.0000          B    DONEJOB
43.0000 NOGET    LA   1,MSG2A
44.0000          OPR  MSG2A,12
45.0000 M        SNAP BEGIN,ENDBUFF
46.0000          B    DONEJOB
47.0000 NOPUT    LA   1,MSG3A
48.0000          OPR  MSG3A,12
49.0000 NOGET    SNAP BEGIN,ENDBUFF
50.0000          B    ENDJOB
51.0000 NOBUFF   LA   1,MSG4A
52.0000          OPR  MSG4A,19
53.0000 O        SNAP BEGIN,ENDBUFF
54.0000 DONEJOB  EOJ
55.0000          DS   0F
56.0000 PRFA     DTFCP TYPE=GT,ERRET=NOGET
57.0000          DS   0F

58.0000 OTPT     DTFCP TYPE=PT,ERRET=NOPUT,DEST=(T,TRM1),NOBAV=NOBUFF
59.0000 MSG1A    DC   CL26'ERROR ISSUING NETREQ MACRO'
60.0000 MSG2A    DC   CL12'ERROR ON GET'
61.0000 MSG3A    DC   CL12'ERROR ON PUT'
62.0000 MSG4A    DC   CL19'NO BUFFER AVAILABLE'
63.0000 CLEAR    ZO#FORMC 0,0
64.0000 COORD1   ZO#COORD 10,30
65.0000 COORD2   ZO#COORD 12,32
66.0000 MSG1     DC   C'YOU USE DICE CHARACTERS'
67.0000 MSG2     DC   C'TO FORMAT YOUR DATA'
68.0000 TXTCHAR  DC   X'0107'
69.0000          DS   0H
70.0000 BUFFER   DC   CL256' '
71.0000 ENDBUFF  DS   0H
72.0000 MSGIN    DC   X'0000'
73.0000          DC   CL5' '
74.0000 TXTIN    DC   CL256' '
75.0000 FINISH   DS   0H
76.0000          END
```

Figure 2-13. DICE Macroinstruction Coding (Part 2 of 3)

Line	Explanation
16-22	Same functions as with hexadecimal DICE (Figure 2-12)
	DICE macroinstructions and decimal representation:
63	Clears screen
64	Moves cursor to row 10, column 30
65	Moves cursor to row 12, column 32
66, 67	Message text

NOTE:

You can specify an ERRET= address, as shown on line 14, only when you don't specify DUSTERR=INLINE in the CCA macroinstruction.

Figure 2-13. DICE Macroinstruction Coding (Part 3 of 3)

2.20. GENERAL STANDARD INTERFACE CONSIDERATIONS

The following considerations should be noted for general operation of standard interface (STDMCP):

- Because the error/control flags in both DTFs are set by you and ICAM, it is mandatory that you clear these fields of all unwanted settings before you issue another GETCP/PUTCP. Unpredictable results will be obtained if this rule is not followed.

- Terminal level queueing with multiple priorities is recommended for all but the simplest networks. Basic networks - only one RDH (TTY or UNISCOPE), no auxiliary device, no MPPS, no disk queueing, no ODNR - are the only ones recommended for line queueing. Small batch configurations will also perform well with line queueing.

- All ICAM terminal queueing is cyclic on a given line. The last terminal to output will be the last considered in looking for new output. All TOP queues (main storage) are considered first, then all highs, mediums, and lows, respectively.

- Computer message waiting (TM#DCMWK) to a UNISCOPE 100 terminal may be included with the text message. ICAM will wait until the terminal responds before sending the text message.

- If a TTY operator terminates with an EOT from the keyboard, a single-character (EOT) text message will be generated and sent to you. A disconnect status will follow at ERRET.

- DTF address and boundary errors will result in a cancel of your program, since ICAM cannot safely return to your program from this error.

- You must eventually execute a CYIELD or perform a non-IRL GETCP/PUTCP to ICAM every time ICAM awakens your program or you will be logically disconnected from ICAM. TYIELDs are not a substitute for CYIELDs unless a CAWAKE is to be executed soon (see the ICAM concepts and facilities, UP-8194 (current version)). This includes line-down and ODNR entries, as well as all SVC returns.

- ICAM is event-driven; therefore, ICAM cannot guarantee full-register environment on every return to a user. For example, if you execute a GETCP, your next return could be a LINE DOWN notice. This notice will have registers 0 and 1 as defined, but all other registers will be from the GETCP.

  If you now do a LNEREL, your next return will probably be inline or the error return from the GETCP. Registers 0 and 1 are as defined, but all other registers are from the LNEREL. You must protect your own environment on all ICAM SVCs.

- Alternate destination, intercept queues, and inhibit input until output features are supported in the STDMCP without MPPS. However, they require CNC5 which is almost 2k bytes versus CNC1 which is 1k bytes.

- Mixed terminal and line queues are supported on the same line. You may select TERM queueing for high priority messages, and LINE queueing for medium and low priority messages. Mixed disk and main storage queueing also is supported.

- Text replacement optionally permits your program to hold a message segment and then later modify the first four bytes of the held segment. This option is useful when device control information is to be supplied on a later message text segment. The text replacement is controlled via bit settings in the DTF. The program sets the TM#DHS bit to hold the message segment, sets the TM#DMS bit to modify the segment, and supplies the DICE to overlay in the TM#DRPL field. Once a segment is held, it can be released in one of two methods:

  1. Issue back to back hold segment requests.

  2. Issue a modify request with the TM#DRPL bit containing zeros.

- When writing a multi-mode basic assembly language program using standard register linkages, you must keep all ICAM macroinstruction calls and error return addresses within the same module. For example, suppose a program consists of two modules with different register sets. Module A contains the NETREQ and network error return; module B contains the PUTCP. If a line error occurs while traffic is being sent from the program to a terminal, ICAM returns control to the program at the network error return (module A) when the program issues a PUTCP (module B). Thus, the program receives control in module A with module B's register set. If you use standard register linkages, the cover register is invalid and the results could be disastrous for the program.

2.21. DECLARATIVE MACROINSTRUCTIONS

The standard interface provides two declarative macroinstructions: DTFCP and DLIST. You code them in the nonexecutable portion of your BAL program.

The DTFCP macroinstruction defines the file table that serves as the interface between ICAM and your program's input and output queues. There are two versions of this macroinstruction. DTFCP with TYPE=PT generates the file table associated with output messages; DTFCP with TYPE=GT generates the file table associated with input messages.

The DLIST macroinstruction specifies a list of terminals, lines, and user programs (locap files) considered collectively to be a single destination.

DTFCP

(Output File, PUTCP Related)

2.21.1. Define the Output File (DTFCP)

Function:

Defines an output file for your program. It is used in conjunction with the PUTCP macroinstruction to send messages to terminals.

Format:

LABEL	△OPERATION△	OPERAND
symbol	DTFCP	TYPE=PT[,NOBAV=symbol] [,DEST={(T,terminal-name) | (P,{process-filename | locap-name}) | (D,distribution-list-name[,U])}] [,LEVEL={LOW | MEDIUM | HIGH | TOP}] [,UNIT=LINE] [,ERRET=symbol]

Label:

symbol
One to four alphanumeric characters that identify this output file DTFCP.

Parameters:

TYPE=PT
Specifies that the file table generated is used by ICAM's Put processing routines.

NOBAV=symbol
Identifies an entry point (label) in your program to which PUTCP processing returns control if no network buffer is available for a message. If omitted, error return is inline. If IRL is set, error return is the completion address. Although specifying NOBAV is optional, we highly recommend that you use it.

DEST=
Identifies the final destination of messages associated with this destination interface.

T
Identifies the final destination as a terminal.

terminal-name
Explicitly identifies the label of the output terminal (TERM) in the network definition.

P
Identifies the final destination as a process or locap file.

{process-filename / locap-name}
Explicitly identifies the label (name) of the process file (PRCS) macroinstruction or the user program locap file (LOCAP macroinstruction) in the network definition.

D
Indicates that final destinations are specified on a distribution list.

distribution-list-name
Is the label of the DLIST macroinstruction that defines the distribution list.

U
Identifies the DLIST as user defined; that is, in a user program rather than in a network definition. This parameter is required when sending messages to destinations included on a user-defined DLIST. (See the ICAM concepts and facilities, UP-8194 (current version) for more details on user-defined DLIST.)

If omitted, indicates the corresponding field (TM#DENA) in the DTFCP table is set before the first PUTCP macroinstruction is executed.

LEVEL=
Further identifies the final destination of a message by specifying the queueing priority of the message.

LOW
Indicates output messages are to be directed to the low priority queue of the specified destination.

MEDIUM
Indicates output messages are to be directed to the medium priority queue of the specified destination.

HIGH
Indicates output messages are to be directed to the high priority queue of the specified destination.

TOP

Indicates output messages are to be directed to the top of the high priority queue of the specified destination.

UNIT=LINE

Indicates each message or message segment is to be marked with a standard end-of-line sequence (1001^{16}) when it is transferred to a network buffer. The sequence will be transformed into an appropriate device-dependent end-of-line sequence when the message is actually transmitted.

If omitted, indicates an entire message or message segment is transferred from your program work area each time a PUTCP is executed.

ERRET=symbol

Identifies an entry point (label) in your program to which PUTCP processing returns control to your program if it detects an error (other than no buffer available).

If you do not specify an ERRET address and Put processing detects an error, control returns to the next instruction following the PUTCP that references this DTFCP. Whether you specify an error address or not, error indicators are set in the DTFCP as described in Table 2-13.

If you elect to use the output delivery notification request feature by specifying TM#DODNR in the TM#DIND field, you must *not* specify this operand. This is because the field in the DTFCP normally used for an error return is used for the 4-byte identifier for ODNR.

NOTE:

Do not use the immediate return line feature (IRL) when you use the output delivery notification feature unless the PUTCP you issue is from island code.

The DTFCP macroinstruction generates input and output file tables, and calls the DSECTs that map them (thus, if you code PRINT GEN, the DSECTs appear in your program listing).

Figure 2-14 describes the destination file table. Tables 2-12, 2-13, and 2-14 describe TM#DEST, the DSECT that provides the labels you can use to check error flags, change destination names, and perform other output message processing. For instance, if you define your interface area as:

```
OUTMSG    DTFCP    TYPE=PT,NOBAV=NOPUT,DEST=T,TRM1,ERRET=PUTER
```

and, when you execute

```
PUTCP     OUTMSG,WRK1
```

an error occurs; control passes to PUTER. There, you can find the specific cause of the error. You code

```
LA      R3,OUTMSG
USING   R3,TM#DEST
```

to map the file table, then test byte TM#DERR using the masks provided in Table 2-13. Thus,

```
TM    TM#DERR,TM#DEIN
```

is the test for an invalid or missing destination name,

```
TM    TM#DERR,TM#DEWA
```

is the test for an invalid work area address, and so on.

BYTE	0	1	2	3	WORD
0	TM#DIND process control indicators		TM#DERR error notification indicators		1
	indicator 1 (TM#DIND)	indicator 2 (TM#DIND+1)	indicator 1 (TM#DERR)	indicator 2 (TM#DERR+1)	
4	output file name (from label) (TM#DNAM or TM#DDST)				2
8	destination routing name (label of TERM, PRCS, LOCAP or DLIST to which message is directed) (TM#DENA)				3
12	TM#DAUX device information		PUTCP processing flags (TM#DDTF)	start/end of message and auxiliary device information (TM#DSEG)	4
	auxiliary device special function (TM#DSPEC)	auxiliary device index (TM#DDVC)			
16	message header address in CCA (PUTCP) working storage (TM#DCHD)				5
20	no buffer available address or IRL completion address (TM#DNBA)				6
24	error return address or ODNR 4-byte message identifier (TM#DERA)				7
28	replacement text (TM#DRPL)				8

NOTE:

Shaded areas are system-supplied parameters; clear areas are program-supplied parameters.

Figure 2-14. Output DTFCP File Table

Table 2-12. Output DTFCP File Table Detailed Field Descriptions (Part 1 of 2)

Byte	Field					Content	Word
	Label*	Type and Length	Set**				
			D	U	P		
0	TM#DIND	XL2			R	Processing flags and indicators; refer to Table 2-13. Indicates status of current message and result of latest PUTCP.	1
2	TM#DERR	XL2			R	Error flags and indicators; refer to Table 2-13. Nonzero only if TM#DIER set in TM#DIND.	
4	TM#DNAM or TM#DDST	CL4	R			Output file DTFCP name (from label field) or destination floating queue name	2
8	TM#DENA	CL4	R	R		Destination name: label of TERM, PRCS, LOCAP, or DLIST to which messages are directed.	3
12	TM#DAUX or	CL2			R	Auxiliary device information for device handler; refer to Table 2-14.	4
12	TM#DSPEC					Auxiliary function code	
13	TM#DDVC					Logical auxiliary device number	
14	TM#DDTF	XL1				Optional flags to control PUTCP processing	
	TM#DUDL		O	O		Destination is defined by a DLIST in user's program	
	TM#DFQR			O		Floating queue request flag	
	TM#DLTR		O			Line transfer requested	
	TM#DDIND					Destination is indirect; message put on queue associated with DNAM.	
	TM#ETOP		O			Top message (priority function codes)	
	TM#EHIG		O			High priority	
	TM#EMED		O			Medium priority	
	TM#ELOW		R			Low priority	
15	TM#DSEG	XL1				Segment definitions	
	TM#DHS			O		Hold segment for modification	
	TM#DMS			O		Modify segment (overlay first four bytes with DICE sequence)	
	TM#DQHLD			O		Queue hold in effect	
	TM#DQRLS			O		Queue release in effect	
	TM#DAFN			O		Set auxiliary device function	
	TM#DBEOF			R		Batch device end-of-file segment	
	TM#DHDR			O		Start of header	
	TM#DTND			R		End of text	
16	TM#DCHD	AL			R	Message header address in CCA; working storage for PUTCP processing. Not addressable by your program.	5

Table 2-12. Output DTFCP File Table Detailed Field Descriptions (Part 2 of 2)

Byte	Field					Content	Word
	Label*	Type and Length	Set**				
			D	U	P		
20	TM#DNBA	AL	O	O		No buffer available (NOBAV) address in your program. Zero if not supplied	6
24	TM#DERA	AL	O	O		ERRET address in your program. Zero if not supplied.	7
28	TM#DRPL	AL				Replacement text. Used with TM#DMS to specify modification to start of segment.	8

* DSECT label is TM#DEST

** D – field normally set by parameters in DTFCP call
U – field can be set by user program before PUTCP
P – field set by software processors
R – required; always set
O – optional

Table 2-13. Error Indicators and Processing Flags in Output DTFCP File Table (Part 1 of 2)

Label		Condition/Cause	Consequences	Go to ERRET?
Byte	Bit*			
TM#DIND	TM#DIRL	Immediate return line	Set by user	
	TM#DODNR	Output delivery notification request	Line status and identifier received in ERRET address	
	TM#DIER	Indicates an error has occurred during PUTCP processing. TM#DERR and TM#DERR+1 should be tested for the type of error.	Refer to specific error flag.	Yes
	TM#DIIL	The TM#DDTF byte of the DTFCP contains an invalid priority or none at all.	PUTCP processing was completed with low priority assumed.	No
	TM#DIMH	A PUTCP was executed when no message was in process, and flag TM#DHDR in byte TM#DSEG was not set.	The contents of your work area were accepted by the MCP as a complete message and transmitted to the specified destination.	No
	TM#DIMA	An incomplete multisegment or multi-line message was aborted in response to a request from your program.	Previous message is discarded; current message is started normally.	No
	TM#DIMP	TM#DTND not set. Indicates a message in process, i.e., PUTCP processing has processed the first or intermediate line of a message.	Message is assumed to be multiple segment or line.	No
	TM#DIIM	Invalid modification. TM#DMS was set but no segment held for modification.	No text replacement. Message is still put.	No

Table 2-13. Error Indicators and Processing Flags in Output DTFCP File Table (Part 2 of 2)

Label		Condition/Cause	Consequences	Go to ERRET?
Byte	**Bit***			
TM#DERR		Error flags are set in TM#DERR and TM#DERR+1 when the PUTCP cannot proceed. If TM#DIER is not set, these two bytes are all zeros.	Message transfer is not initiated or completed; refer to specific error conditions for further consequence.	
	TM#DEIN	Invalid/missing destination name: TM#DENA does not match the label of any TERM, PRCS, or DLIST in the CCA.	DTFCP is unusable.	Yes
	TM#DEWA	Work area address or size invalid: not within your program boundaries.	PUTCP with valid work area address can be performed.	Yes
	TM#DENB	Network buffers not available for header (first segment/line, whole message).**	Control goes to NOBAV address.	No
	TM#DEIA	Header address (TM#DCHD) invalid: your program has altered header address working storage in the DTFCP.	Before next PUTCP, TM#DCHD must be set to zero. If error occurred in the middle of a message, the previous segments/lines are lost.	
TM#DERR+1	TM#DEDU	Invalid DLIST entry; a destination name within a DLIST does not match any TERM or PRCS name in the network. Generation of a new network may be required to correct the error.	Message queued for transmission to all destinations on the DLIST from the second up to (not including) the invalid one. Message is not transmitted to the first, the invalid, or any subsequent destinations.	Yes
	TM#DIRLE	Immediate return line error flag set. Indicates that IRL was requested without supplying a completion address in TM#DNBA.	No data is transferred from the work area of your program. You may supply a completion address and reissue the PUTCP.	Yes
	TM#DODNE	Output delivery notice was requested without supplying an address in TM#DERA.	No data is moved from the work area of your program. Your program may enter an address in TM#DERA and execute another PUTCP with TM#DODNR set.	Yes
	TM#DIRLE and TM#DODNE	Indicates that a PUTCP was issued with both IRL and output delivery notice requested. The put is invalid.	No data is moved from your program. You may reissue the PUTCP with only one flag set.	Yes

* Bit label positions in each byte do not indicate value. Actual values are contained in the DTFCP.

** Flag TM#DENB indicates no network buffer is available for the current message, segment, or line. If the PUTCP was for the transfer of a complete message, the first segment, or the first line of a message, the data in your work area will be intact. The work area prefix value, however, may have been altered. When buffers are available, you must restore the original work area prefix byte count and execute a PUTCP to reinitiate transfer of the message.

If the current PUTCP is for the transfer of a segment or line other than the first, previously transferred segments or lines are not affected. In this case, reinitiation of the message transfer should begin with the current segment or line.

Your program can assist in recovery from a no-buffer condition by executing GETCPs to dequeue input messages, thereby freeing buffers for use. Frequent occurrence of this error may indicate a need to generate a new network making an adjustment to the number and/or size of buffers.

Flag TM#DENB may also be set by ICAM when the length of a message to a batch device exceeds the space available in a network buffer. Generation of a new network, increasing the buffer size, is indicated.

Table 2-14. Auxiliary Device and Special Function Specifications in Output DTFCP File Table

Label	Function	Remarks
TM#DADBS	Backspace tape cassette one block	TCS on UNISCOPE display terminal only
TM#DADPF	Print form	UTS 400 only
TM#DADSR	Search tape cassette	TCS on UNISCOPE display terminal only; message text is search argument.
TM#DADRA	Report current address from tape cassette	TCS on UNISCOPE display terminal only
TM#DADRD	Read a block (magnetic or paper tape)	TCS on UNISCOPE display terminal or DCT 524. Paper tape reader on DCT 500 or TTY
TM#DADWR	Write a block (magnetic or paper tape)	TCS on UNISCOPE display terminal or DCT 524. Paper tape reader on DCT 500 or TTY
TM#DADXA	Transfer all	UTS 400 only
TM#DADXC	Transfer changed	UTS 400 only
TM#DADXV	Transfer variable	UTS 400 only
TM#DCSHT	Send halt	BSC terminal
TM#DFSMW	Light computer message waiting and sound audible alarm	DCT 1000
TM#DFSMW	Send message-waiting or sound bell	UNISCOPE display terminal, UTS 400/UTS 4000, TTY, DCT 500, DCT 524, or DCT 475
TM#DPRE	Prevent nonsignificant space suppression	UNISCOPE display terminal only (primary or auxiliary)
TM#DTAT	Transparent	UNISCOPE aux auxiliary device
TM#DCRVI	RVI (BSC)	BSC terminal
TM#DCTRN	Transparent BSC	BSC terminal
TM#DADWR ++ TM#DTAT	Load program on diskette	UTS 400
TM#DCNCH	Send no change indicator	1004 slave handler
TM#DCRDF	Send read function. Initiate output.	1004 slave handler
TM#DCSOL	Send offline function	1004 slave handler
TM#DCINI	Send ready and site-id; start polling	1004 slave handler
TM#DCPRA	Send abort print	1004 slave handler
TM#DCPUA	Send abort punch	1004 slave handler
TM#DCBCN	Send BSC connect sequence	BSC terminal
TM#DCMWK	Send computer message waiting	UNISCOPE
TM#DCBDC	Send BSC DLE EOT	BSC terminal

NOTES:

1. All labels are set in TM#DSPEC.

2. Labels starting with TM#DAD denote auxiliary device functions. Other labels apply to special I/O functions and do not require the specification of an auxiliary device function.

DTFCP

(Input File, GETCP Related)

2.21.2. Define the Input File (DTFCP)

Function:

Defines an input file for your program. It is used in conjunction with the GETCP macroinstruction to retrieve messages from the queues of a process file, locap file, or a terminal's input message queue.

Format:

LABEL	△OPERATION△	OPERAND
symbol	DTFCP	TYPE=GT [,UNIT=LINE] [,LEVEL=(LOW / MEDIUM / HIGH / AVAIL)] [,NOMAV=symbol] [,ERRET=symbol] [,DATIME=YES] [,NOTLST=symbol] [,ODN=YES]

Label:

symbol
One to four alphanumeric characters that identify this input file DTFCP. This label must match the label of the corresponding PRCS, LOCAP, or TERM macroinstruction in the network definition.

Parameters:

TYPE=GT
Specifies that the file table (see Figure 2-2) generated is used by ICAM's Get processing routines.

UNIT=LINE
Indicates that a single line of a message is to be transferred to your program work area each time a GETCP macroinstruction referencing this file is executed. The work area must be large enough to accommodate the largest line.

If omitted, an entire message or a message segment is transferred into the work area each time a GETCP macroinstruction is executed.

LEVEL=
Indicates, by priority level, which of the possible three queues associated with the process file/locap file is to be accessed for messages. It is used during processing of a GETCP macroinstruction specifying a priority level. If MPPS is not supported, only low level is applicable.

LOW
Indicates the low priority queue of the process file is to be the queue accessed for a message.

MEDIUM
Identifies the medium priority queue as the queue to be accessed.

HIGH
Identifies the high priority queue as the queue to be accessed.

AVAIL
Indicates the queues are accessed in descending order of priority.

NOMAV=symbol
Identifies an entry point (label) in your program to which GETCP processing returns if no message is available. To try again, the program must execute another GETCP macroinstruction.

If omitted, your program is suspended until a message becomes available. If IRL is in effect, this address is the completion address activated when a message becomes available.

ERRET=symbol
Specifies an entry point (label) in your program to which GETCP processing returns if it encounters an error condition other than no message available.

If omitted, processing control returns to the next instruction in the program. In either case, error indicators are set in the associated DTFCP file table.

Figure 2-15 describes the input DTFCP file table. Tables 2-15, 2-16, and 2-17 describe TM#PRCS, the DSECT that provides the labels you can use to check error flags, change message originator names, and perform other input message processing.

For example, if you define your interface area as:

```
INPT DTFCP TYPE=GT,NOMAV=NOGET,ERRET=GETERR
```

and, when you execute

```
GETCP INPT,WRK2
```

an error occurs, control passes to GETERR. There, you can find the specific cause of the error. You code

```
LA      R2,INPT
USING   R2,TM#PRCS
```

to map the input file table, then test TM#PERR using the masks provided in Table 2-15. Thus,

```
TM   TM#PERR,TM#PEIN
```

is the test for an invalid or missing file name;

```
TM   TM#PERR,TM#PEWA
```

is the test for an invalid work area address, and so on.

DATIME=YES
: Places the date and time of the message in the DTFCP.

NOTLST=symbol
: Specifies the address of a list of process files, locap files, or terminal names. Each list entry must be four bytes.

ODN=YES
: Specifies that a word is to be generated in the DTFCP for the output delivery notification identifier if the ODN option is selected on the GETCP. (See 2.10.)

BYTE	0	1	2	3	WORD
0	TM#PIND process control indicators		TM#PERR error notification indicators		1
	indicator 1 (TM#PIND)	indicator 2 (TM#PIND+1)	indicator 1 (TM#PERR)	indicator 2 (TM#PERR+1)	
4	process file, locap file, or terminal name (from DTFCP or user program supplied) (TM#PNAM)				2
8	name (label) of source message originator (TM#PSRC)				3
12	reserved		optional GETCP process control flags (TM#PDTF)	user program dynamic message control (TM#PSEG)	4
16	message header address in CCA (GETCP) working storage (TM#PCHD)				5
20	no message available address or IRL completion address (TM#PNMA or TM#PCMPA)				6
24	error return address (TM#PERA)				7
28	TM#PAUX device information		number of bytes remaining in current message (TM#PRSG)		8
	auxiliary device/ special function (TM#PSPEC)	auxiliary device index (TM#PDVC)			
32	address of input message notification list (notlst) (TM#PAINT)				9
36	date (when using date/time stamp) (TM#PDATE)				10
40	time (when using date/time stamp) (TM#PTIME)				11
44	TM#PTIME (cont)	not used			12
48	output delivery notification identifier (TM#PODNN)*				13

* Word 13 is generated only if the ODN=YES parameter is specified.

NOTE:

Shaded areas are system-supplied parameters; clear areas are user-supplied parameters.

Figure 2–15. Input DTFCP File Table

Table 2-15. Input DTFCP File Table Detailed Field Descriptions (Part 1 of 2)

Byte	Field					Content	Word
	Label*	Type and Length	Set**				
			D	U	P		
0	TM#PIND	XL2			R	Processing flags and indicators; see Table 2-17. Status of current message and result of latest GETCP.	1
2	TM#PERR	XL2			R	Error flags and indicators; see Table 2-17. Nonzero only if TM#PIER set in TM#PIND.	
4	TM#PNAM	CL4	R			Process file, locap file, or terminal name. Specified from the DTFCP label field or within the user program. Corresponds to PRCS, LOCAP, or TERM macro-instruction in CCA network definition.	
8	TM#PSRC	CL4			R	Name (label) of source. Normally a terminal name; if TM#PISR is set in TM#PIND, the source is a user program, not a terminal.	3
12	Reserved	H					
14	TM#PDTF	XL1	O	O		Optional flags to control GETCP processing	
	TM#PDTSP					Date/time request	
	TM#PINPN					Input notice request	
	TM#PCSD					Reserved for COBOL MCS	
	TM#PLTR					Each GETCP delivers one line to work area. Set by UNIT-LINE	
	TM#EAVL					Available message. Controls level of queue from which message is taken. See LEVEL parameters of DTFCP macroinstruction.	
	TM#EHIG					High priority	
	TM#EMED					Medium priority	
	TM#ELOW					Low priority	
15	TM#PSEG	XL1		O		Contains two message processing bits. Used when getting incomplete messages because work area is too small or when not accessing segments/lines in normal sequence.	
	TM#PNHD					Get new header; skip remainder of current message. GETCP yields next message or first segment/line of next message.	
	TM#PSHD					Start header; revert processing to beginning of current message. GETCP yields first segment/line of current message.	
16	TM#PCHD	A			R	Message header address in CCA, working storage for GETCP processing. Not addressable by user program.	5

Table 2-15. Input DTFCP File Table Detailed Field Descriptions (Part 2 of 2)

Byte	Field					Content	Word
	Label*	Type and Length	Set** D	Set** U	Set** P		
20	TM#PNMA or TM#PCMPA	A	O	O		No message available (NOMAV) address in user program or IRL completion address. Zero if not supplied.	6
24	TM#PERA	A	O	O		ERRET address in user program. Zero if not supplied.	7
28	TM#PAUX TM#PSPEC	H			O O	Flags for special I/O functions, conditions, and auxiliary device input. See Table 2-16.	8
29	TM#PDVC				O	Auxiliary device number if message received from auxiliary device (01_{16} thru OC_{16}). Valid only if TM#PAUXD is set in TM#PSPEC.	
30	TM#PRSG	H			R	Number of remaining segments in current message	
32	TM#PAINT TM#PLEN	A		O		Address input notice list Process file DSECT length	9
Following fields present with date/time stamp							
36	TM#PDATE	PL4		O		Date	10
40	TM#PTIME	PL5		O		Time	11,12
45	 TM#PDLEN	XL3				Not used Length with date/time	
48	TM#PODNN	CL4			O	Output delivery notice (ODN) information	13

* DSECT label is TM#PRCS

** D – field normally set by parameter in DTFCP macro
U – field can be set by user program before GETCP
P – field set by software processor
R – required; always set
O – optional

Table 2-16. Auxiliary Devices and Special Function Flags in Input DTFCP File Table (Part 1 of 2)

Label*	Function/Condition	Remarks
TM#PFRMW or TM#PCMWK	Message-waiting or bell	Interactive terminal
TM#PFKY1 or TM#PCFK1	Function key 1	UNISCOPE/UTS 400/ UTS 4000 terminal
TM#PFKY2 or TM#PCFK2	Function key 2	UNISCOPE/UTS 400/ UTS 4000 terminal
TM#PFKY3 or TM#PCFK3	Function key 3	UNISCOPE/UTS 400/ UTS 4000 terminal
TM#PFKY4 or TM#PCFK4	Function key 4	UNISCOPE/UTS 400/ UTS 4000 terminal
TM#PFKY5	Function key 5	UTS 400/UTS 4000 terminal
TM#PFKY6	Function key 6	UTS 400/UTS 4000 terminal
TM#PFKY7	Function key 7	UTS 400/UTS 4000 terminal
TM#PFKY8	Function key 8	UTS 400/UTS 4000 terminal
TM#PFKY9	Function key 9	UTS 400/UTS 4000 terminal
TM#PFK10	Function key 10	UTS 400/UTS 4000 terminal
TM#PFK11	Function key 11	UTS 400/UTS 4000 terminal
TM#PFK12	Function key 12	UTS 400/UTS 4000 terminal
TM#PFK13	Function key 13	UTS 400/UTS 4000 terminal
TM#PFK14	Function key 14	UTS 400/UTS 4000 terminal
TM#PFK15	Function key 15	UTS 400/UTS 4000 terminal
TM#PFK16	Function key 16	UTS 400/UTS 4000 terminal
TM#PFK17	Function key 17	UTS 400/UTS 4000 terminal
TM#PFK18	Function key 18	UTS 400/UTS 4000 terminal
TM#PFK19	Function key 19	UTS 400/UTS 4000 terminal
TM#PFK20	Function key 20	UTS 400/UTS 4000 terminal
TM#PFK21	Function key 21	UTS 400/UTS 4000 terminal
TM#PFK22	Function key 22	UTS 400/UTS 4000 terminal
TM#PCRRD	Received read during output	1004 terminal
TM#PCRHT	Received halt	1004 terminal
TM#PCRHV	Received halt, go voice (1004) or parity error (TTY/DCT 500)	1004 terminal or TTY/DCT 500 Series terminal

Table 2-16. Auxiliary Devices and Special Function Flags in Input DTFCP File Table (Part 2 of 2)

Label*	Function/Condition	Remarks
TM#PCRER	Received end read	1004 terminal
TM#PCRDY/ TM#PCRDV	Received ready	1004 terminal
TM#PCOFF	Received offline	1004 terminal
TM#PCAPR	Received abort print	Batch terminal
TM#PCAPU	Received abort punch	Batch terminal
TM#PCRVI	Received RVI	BSC terminal
TM#PXEOT	Operator pressed hang-up key	UTS 400/UTS 4000 terminal
TM#PPOC	Firmware completed power-on confidence test	UTS 400/UTS 4000 terminal
TM#PCBRK	Break or interrupt received	TTY/DCT 500 Series terminal

* All labels set in TM#PSPEC.

Table 2-17. GETCP Error Indicators and Processing Flags in Input DTFCP File Table (Part 1 of 2)

Label		Condition/Cause	Consequences	Go to ERRET?
Byte	Bit*			
TM#PIND	TM#PIRL	Immediate return line	Set by user; never cleared by ICAM	
	TM#PIER	Message not obtained. Check TM#PERR and TM#PERR+1 for exact cause.	Depends on type of error	Yes
	TM#PIIL	Priority level invalid. TM#PDTF of the DTFCP contained an invalid or zero priority value.	GETCP processing was done with AVAIL assumed.	No
	TM#PISR	Message entered system from a user program, not a terminal.	None: informational only	No
	TM#PBEOF	Indicates an end of file was detected in the input message from a batch device.	None: informational only	No
	TM#PIMD	Work area did not accommodate entire text of message.	Additional GETCPs must be done to get remainder of message.	No
	TM#PIHD**	Start of new message; indicates the message header has been transferred to your work area.	Informational	No
	TM#PIET**	End of present message; ICAM indicates the data at the end of the message has been transferred to your work area. The message is complete.	Next GETCP will obtain a new message.	No

Table 2-17. GETCP Error Indicators and Processing Flags in Input DTFCP File Table (Part 2 of 2)

Label		Condition/Cause	Consequences	Go to ERRET?
Byte	Bit*			
TM#PIND+1	TM#PIDL	Data lost. Your program has canceled a message. Previous message was truncated because TM#PNHD was set when GETCP was executed. (Refer to TM#PSEG.)	Remainder of previous message is not available.	No
	TM#PDTER	Invalid date/time request: date/time stamping requested, but CCA configured doesn't provide this option.		No
TM#PERR		These bits are set only if TM#PIER was set in TM#PIND.		
	TM#PEIN	Invalid end user specification: an error occurred while attempting to get a message from the name specified in TM#PNAM (TM#PNAM doesn't match label of any end user in CCA; no valid QCT configured for end user specified; inconsistent message information between QCT and DTFCP).	DTFCP is unusable. No data is moved into your work area. Correct the invalid file name or CCA generation and reissue the GETCP.	Yes
	TM#PELR	Line request not available.	Return to user	Yes
	TM#PEWA	Work area size = 0, or work area ending address not within your program boundaries or not on half-word boundary.	Message not transferred	Yes
	TM#PENA	Invalid notice list address: address specified in TM#PAINT is not within bounds of user region.	Abort GETCP	Yes
	TM#PEIA	Message header address (TM#PCHD) invalid; your program has altered header address working storage in the DTFCP.	Before next GETCP, TM#PCHD must be set to zero. If error occurred in the middle of a message, the remainder of the message is lost.	Yes
TM#PERR+1	TM#PEID	Invalid deferral request. A deferred GETCP or input message notification was issued against an end user that already has a deferral request outstanding.	The second deferral request was not processed.	Yes

* Bit label positions in each byte do not indicate value. Actual values are contained in the DTFCP.

** IHD and IET are both set if a complete message was transferred into the work area.

DLIST

2.21.3. Define a Distribution List (DLIST)

Function:

The DLIST macroinstruction allows you to specify one name that refers to several destinations. The distribution list may reference terminals, process files, user programs (locap files), or other distribution lists. Messages are output via a PUTCP macroinstruction that references the DLIST. This is equivalent to a group of Put instructions that reference every destination in the DLIST; that is, a separate transmission is performed to every terminal, process file, LOCAP, or DLIST. Note that a DLIST cannot be nested more than once, i.e., a DLIST referenced by a DLIST cannot reference a third DLIST.

Format:

LABEL	△OPERATION△	OPERAND
dlist-name	DLIST	$destination_1$,$destination_2$....

Label:

dlist-name
Is the 1- to 4-character label of the distribution list generated by this macroinstruction. This label is required.

Positional Parameters 1–n:

$destination_1$,$destination_2$...
Identifies the terminals, process files, DLISTs, or LOCAPs as the destinations for a message.

NOTES:

1. *An implied DLIST (a PUTCP to either a network or a line name) is restricted.*

2. *A DLIST with line queueing is not supported.*

3. *If you use distribution lists in your program only, you must include at least one DLIST macroinstruction in your network definition. Refer to the ICAM network definition and operations user guide, UP-8947 (current version).*

Examples:

```
1       10    16
-------------------------------------------
DLT1    DLIST TER1,TER2,DLT2,PFL1,PFL2,DLT3
DLT2    DLIST TER3,PFL3,LOC1,LOC2,TER4
DLT3    DLIST TER4,LOC3
```

2.22. IMPERATIVE MACROINSTRUCTIONS

The imperative macroinstructions can be grouped into the categories shown in Table 2-18.

Table 2-18. Imperative Macroinstructions

Macroinstruction	Category
NETREQ NETREL LNEREQ LNEREL NATTACH NDETACH SESCON TRMREL	Acquiring and releasing communications facilities
CYIELD CAWAKE GAWAKE	Relinquishing and acquiring communications control
GETCP PUTCP	Sending and receiving messages
CCACPY QCLEAR QDEPTH QHOLD QRELSE QTRANS RELEASM TRMREP	Displaying and altering network status

2.22.1. Acquiring and Releasing Communications Facilities

You use these macroinstructions to:

- acquire or release a dedicated network;
- activate or release lines or terminals in a dedicated network;
- attach or detach your program to or from a global network; and
- open and close global network sessions.

In a *dedicated network* system, communications facilities are acquired and released by the following macroinstructions issued by your program:

- NETREQ

  Activates a dedicated communications network and allows you to activate a single line on that network or all lines on the network.

- LNEREQ

  Initiates the linkage between a channel and a line described in a dedicated network.

- LNEREL

  Terminates the linkage between a channel and a line described in a dedicated network definition.

- NETREL

  Releases a dedicated network.

In a *global network* system, your program uses the following macroinstructions to attach itself to a global network, to open and close a session with the desired end users, and to detach itself from the network when it is finished.

- NATTACH

  Initiates the linkage between your program and a global network and identifies your program to the global network.

- NDETACH

  Detaches your program from a global network.

- SESCON

  Opens, closes, accepts, rejects, aborts, or confirms the closing of a dynamic session.

- TRMREL

  Disconnects a terminal from a circuit-switched public data network.

You can access the parameter tables generated by these macroinstructions by expanding TN#DSECT.

NETREQ

2.22.1.1. Activate Network (NETREQ)

Function:

Activates a previously defined network. Used with dedicated networks only.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	NETREQ	network-name ,ERRET=symbol [,PASSWORD=password] [,LNEREQ={(YES)/(NO)}] [,RESTART=YES] [,FATAL={(YES)/(NO)}] [,MF=({(C/D/N)[,{prefix-code/*/&}]}/{(E,{parameter-list-label/(1)})/L})]

Parameters:

network-name
Identifies the network to be activated. The network name is defined by the CCA macroinstruction.

ERRET=symbol
Specifies a *required* symbolic address in your program where you want ICAM to return control for the following conditions:

1. communications line down or unidentified terminal on line;
2. output delivery notification request (ODNR); or
3. errors in execution of a macroinstruction other than GETCP, PUTCP, or GAWAKE.

If you specify DUSTERR=INLINE on the CCA macroinstruction when you define your ICAM network definition, all execution errors (condition 3) except for GETCP, PUTCP, and GAWAKE return inline. Errors listed in conditions 1 and 2 always return to the error return address you specify in the ERRET= operand. See Table 2-24.

PASSWORD=password

One to eight characters that identify the network password. This parameter is required unless no password was specified when the network was generated.

LNEREQ=

YES

Line request functions are automatically executed for all lines defined in the network.

NO

No line request functions are initiated. Instead, you issue a LNEREQ macroinstruction for each line to be activated.

FATAL=

YES

All network request (NETREQ) errors (TQ#NER1 and TQ#NER2) are treated as fatal errors. The network is not opened or initialized.

NO

Network request (NETREQ) errors reported in TQ#NER2 are not treated as fatal errors, and the network is opened and initialized with errors. Any error in TQ#NER1, regardless of the FATAL keyword, is always fatal and the network is not opened or initialized.

NOTES:

1. *TU#JERR (journal file initialization error), reported in TQ#NER2, is always treated as a fatal error, and the network is not opened or initialized.*

2. *To determine which lines have failed, issue a separate LNEREQ for each line. Lines with any errors reported in TQ#NER2, other than TU#DLAR, are the problem lines for NETREQ.*

RESTART=YES

Specifies that a warm restart is to be performed on all disk queues associated with the named network. This causes all queues to be refreshed to the last point at which a queue checkpoint was taken and the queues placed on hold.

If omitted, a normal NETREQ is performed initializing all queues as empty.

NOTES:

1. *This parameter should not be used the first time a network is requested because there is no information on disk with which to refresh the queues.*

2. *When specified for a network with all main storage queues, the parameter is meaningless and is ignored.*

MF=

Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Example:

```
1         10    16
          NETREQ NET1,ERRET=NET1ERRS,PASSWORD=XXXXXXXX,LNEREQ=NO
          NETREQ TNET,ERRET=ERRHND
          NETREQ NET1,ERRET=NET1ERRS
```

2.22.1.1.1. NETREQ Error Processing

When you specify DUSTERR=INLINE in the CCA macroinstruction, regardless of whether an error is detected during macroinstruction processing, control returns inline to the location following the NETREQ macroinstruction. When you don't specify DUSTERR=INLINE and an error occurs, control is returned at the specified NETREQ ERRET= address. See 2.12 for details.

Register 1 contains the address of the NETREQ parameter table. Register 0 contains:

- zeros in bytes 0 and 1;
- TQ#NER1 error codes in byte 2; and
- TQ#NER2 error codes in byte 3.

If you specify LNEREQ=YES, and a line error occurs while initializing one or more lines, you can control whether the network is opened and initialized through the FATAL keyword.

Table 2–19 describes the error conditions that may be detected during the execution of this macroinstruction.

Table 2-19. NETREQ Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#NER1	TU#DDTAE	NETREQ table outside user region	Control is always returned inline following this request. The request is not honored.
	TU#DDTLE	NETREQ table length incorrect	
	TU#DDTBE	NETREQ table not full-word aligned	
	TU#DERRA	Error return address not in user region.	
	TU#DPASS	Invalid password; password does not match password on CCA macroinstruction	When you specify DUSTERR=INLINE, control is returned inline following this request. When you don't specify DUSTERR=INLINE, control is returned at the address specified in the ERRET operand of the NETREQ macroinstruction. The request is not honored.
	TU#DDSKE	Disk error opening a file	
	TU#DDSKA	Attach error occurred - disk queueing	
	TU#DDSKR	A disk error occurred while reading a file - disk queueing	
	TU#DRAN	Network requested already active	
	TU#DDSKF	File error - disk queueing (file characteristics do not match those in CCA). See note.	
	TU#DUSAT	User saturation - ICAM user slot unavailable to log in user	
	TU#DCSAT	CCA ARP saturation; unable to acquire ARP from CCA to process request	
	TU#DGBL	Job other than GUST attempted NETREQ of global CCA	
TQ#NER2	TU#DNLLT	One or more lines in the network could not be mapped to a physical port. For workstations, the workstation to be linked to ICAM was not physically present.	When you specify DUSTERR=INLINE, control is returned in line following this request. The request is or is not honored, depending on the FATAL=keyword specification. If errors occur for several lines, these flags may be ORed together.
	TU#DNCCT	No available CA tables for one or more lines	
	TU#DLNNE	Line name specified nonexistent	
	TU#DLCAE	CA initialization error	
	TU#DLAR	Line name specified already active. For workstations, the workstation to be linked to ICAM was physically present but not available.	
	TU#DIAL	Autodial line could not be dialed.	
	TU#JERR	Journal file initialization error	When you specify DUSTERR=INLINE, control is returned inline following this request. The request is not honored.

NOTE:

User must decide if warm restart is desired.

- If warm restart is not required, reinitialize the disk queueing file.
- If warm restart is required, determine if buffer size in network was changed since disk queueing file was created.
  - If changed, restore buffer size in network and regenerate.
  - If not changed, user is probably opening a nondisk queueing file.

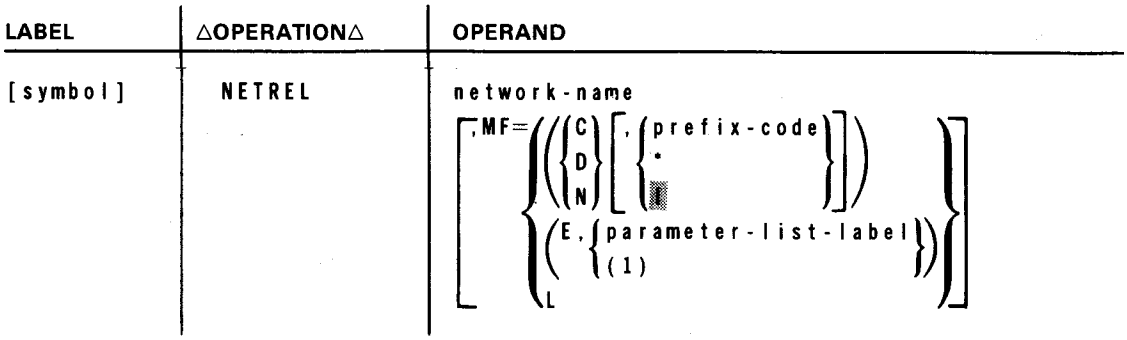
NETREL

2.22.1.2. Release Network (NETREL)

Function:

Releases the facilities comprising a communication network. The release occurs after validation of the NETREL macroinstruction call. Messages remaining on main storage queues are lost after NETREL is executed. Messages remaining on disk queues are not lost unless a NETREQ macroinstruction without the RESTART=YES parameter is initiated. To avoid losing any messages, include a QDEPTH macroinstruction or a time delay loop in your programming.

Format:



Parameters:

network-name
Specifies the network to be released. The network name is defined by the CCA macroinstruction.

MF=
Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Example:

```
1         10    16
-------------------------------------
          NETREL NET1
```

You can use the QDEPTH macro to avoid loss of messages. ICAM checks the message count in all of the end user message queues. When the message count is zero for all queues, ICAM has finished processing all of the messages queued, and you can safely release the network. When using QDEPTH, be careful not to get into a loop and waste CPU time. If a message count shows that ICAM is still processing messages, return control to ICAM by performing a SETIME (of at least 100 milliseconds) macroinstruction.

2.22.1.2.1. NETREL Error Processing

When you specify DUSTERR=INLINE in the CCA macroinstruction and an error occurs during macroinstruction processing, control returns inline to the address following the NETREL macroinstruction. If you do not specify DUSTERR=INLINE, errors are returned at the specified NETREQ ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the NETREL parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#NER1 error codes in byte 2.

Table 2-20 describes the error conditions that may be detected during the execution of this macroinstruction.

Table 2-20. NETREL Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#NER1	TU#DDTAE	NETREL table outside user region	When you specify DUSTERR=INLINE, control is returned inline following this request. When you don't specify DUSTERR=INLINE, control is returned to the address specified in the ERRET operand of the NETREQ macroinstruction. The request is not honored.
	TU#DDTBE	NETREL table not full-word aligned	
	TU#DDTLE	NETREL table length incorrect	
	TU#DFE	NETREL table flags incorrect	

NOTES:

1. The DSECT for this table is TQ#DSCTS.
2. The TQ#NER2 error byte does not apply to NETREL and always contains a zero.
3. Your program is canceled with error code 450 when:
   - you issue a NETREL without a previous NETREQ; or
   - you issue a NETREL after a previous NETREL.

LNEREQ

2.22.1.3. Initiate Linkage (LNEREQ)

Function:

Initiates the required linkage between a channel and a line described in a communications network. This macroinstruction affects all the terminals on a line as described in the network definition. On polled lines, LNEREQ initiates the necessary polling procedures.

NOTE:

The functions performed by this instruction can be initiated as a parameter to the NETREQ macroinstruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	LNEREQ	line-name [,MF=({C,D,N}[,{prefix-code,*}]) / (E,{parameter-list-label,(1)}) / (L)]

Parameters:

line-name
Identifies the line to be linked. This name must be the same as the label of the associated LINE macroinstruction.

MF=
Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Example:

```
1         10    16
          LNEREQ LNE1
```

2.22.1.3.1. LNEREQ Error Processing

When you specify DUSTERR=INLINE in the CCA macroinstruction and an error occurs during macroinstruction processing, control returns inline to the address following the LNEREQ macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the LNEREQ parameter table. Register 0 contains:

- zeros in bytes 0 and 1;
- TQ#LER1 error codes in byte 2; and
- TQ#LER2 error codes in byte 3.

Table 2-21 describes the error conditions that may be detected during the execution of this macroinstruction.

Table 2-21. LNEREQ Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#LER1	TU#DDTAE	LNEREQ table outside user region	When you specify DUSTERR=INLINE, control is returned inline following this request. When you don't specify DUSTERR=INLINE, control is returned at the address specified in the ERRET operand of the NETREQ macroinstruction. This request is not honored.
	TU#DDTBE	LNEREQ table not full-word aligned	
TQ#LER2	TU#DLNNE	Line name specified nonexistent	
	TU#DLAR	Line name specified already active	
	TU#DNLLT	Line cannot be mapped; no available port	
	TU#DNCCT	No available communications adapter (CA) subsystems available for this line	
	TU#DLCAE	CA initialization error	
	TU#DIAL	Autodial line could not be dialed	

NOTE:

The DSECT for this table is TQ#DSCTS.

LNEREL

2.22.1.4. Terminate Linkage (LNEREL)

Function:

Terminates the linkage between a channel and a line described in a communications network. Termination occurs after validation of this macroinstruction call. Any output messages not processed at the time of termination remain on the output queues until the line is reopened via a LNEREQ macroinstruction call.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	LNEREL	line-name [,MF=({(C,D,N)[,{prefix-code,*,1}]},{E,{parameter-list-label,(1)}},L)]

Parameters:

line-name
: Identifies the line to be terminated. This name must be the same as the label of the associated LINE macroinstruction.

MF=
: Identifies a parameter list whose address can be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Example:

```
1         10    16
-------------------------------------------
          LNEREL LNE3
```

2.22.1.4.1. LNEREL Error Processing

When you specify DUSTERR=INLINE in the CCA macroinstruction and an error occurs during macroinstruction processing, control returns inline to the address following the LNEREL macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the LNEREL parameter table. Register 0 contains:

- zeros in bytes 0 and 1;
- TQ#LER1 error codes in byte 2; and
- TQ#LER2 error codes in byte 3.

Table 2-22 describes the error conditions that may be detected during the execution of this macroinstruction.

Table 2-22. LNEREL Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#LER1	TU#DDTAE	LNEREL table outside user region	When you specify DUSTERR=INLINE, control is returned inline following this request. When you don't specify DUSTERR=INLINE, control is returned at the address specified in the ERRET operand of the NETREQ macroinstruction. The request is not honored.
	TU#DDTBE	LNEREL table not full-word aligned	
	TU#DDTLE	LNEREL table length incorrect	
	TU#DFE	LNEREL table flags incorrect	
TQ#LER2	TU#DLNNE	Line name specified does not exist	
	TU#DLAR	Line name specified has already been released or was never requested.	

NOTE:

The DSECT for this table is TQ#DSCTS.

NATTACH

2.22.1.5. Attach Network (NATTACH)

Function:

Attaches user to a global network.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	NATTACH	network-name ,APPS=name ,ERRET=symbol [,PASSWORD=password] [,MF=({D,C,N}[,{prefix-code,*,I}]) [,MF=(E,{parameter-list-label,(1)})] [,MF=L]

Parameters:

network-name
Specifies the name of the global network.

APPS=name
Identifies a 1- to 4-character locap file name that is registered in the global network. This allows an applications program to be addressed in a similar fashion to lines, terminals, and process files.

ERRET=symbol
Specifies a *required* symbolic address in your program where you want ICAM to return control for the following conditions:

1. output delivery notification request (ODNR); or

2. errors in execution of a macroinstruction other than GETCP, PUTCP, GAWAKE, SESCON.

If you specify DUSTERR=INLINE on the CCA macroinstruction when you define your ICAM network definition, all execution errors (condition 2) except for GETCP, PUTCP, GAWAKE, and SESCON return inline. ODNR errors always return to the error return address you specify in the ERRET= operand. See Table 2-24.

NOTE:

A program in a global environment, which must use the NATTACH macroinstruction instead of NETREQ, cannot control communications lines. Therefore, line-down notifications go to GUST rather than to your program's ERRET address, as they do for programs in a dedicated environment.

PASSWORD=password
: One to eight characters that identify the network password. This parameter is required unless no password was specified when the network was generated.

MF=
: Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Example 1:

```
NETST    NATTACH GBLA,ERRET=NETERR,APPS=CUPA
```

Example 2:

```
         LA      1,NAT1
         NATTACH MF=(E,(1))
           .
           .
           .
         LA      1,NAT2
         NATTACH MF=(E,(1))
           .
           .
           .
         EOJ
NAT1     NATTACH GBLA,APPS=CUP1,MF=L
NAT2     NATTACH GBLB,APPS=CUP2,MF=L
```

Example 3:

```
         NATTACH MF=(E,NAT3)
           .
           .
           .
         EOJ
NAT3     NATTACH GBLC,APPS=CUP3,MF=L
```

2.22.1.5.1. NATTACH Error Processing

When you specify DUSTERR=INLINE in the LOCAP macroinstruction and an error occurs during macroinstruction processing, control returns inline to the location following the NATTACH macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NATTACH ERRET= address. However, we recommend that you code your programs for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the NATTACH parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#NER1 error codes in byte 2.

Table 2-23 describes the error conditions detected during the execution of this macroinstruction.

Whenever the status of the permanent virtual circuit (pvc) changes, you receive the up or down flag and the permanent virtual circuit number in register 0. Byte 1 contains the status byte (X'41' for pvc up; X'42' for pvc down). Bytes 2 and 3 contain the virtual circuit number. Register 1 contains the 4-character name of the remote locap.

Table 2-23. NATTACH Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#NER1	TU#DDTAE	NATTACH table outside user region	When you specify DUSTERR=INLINE, control is returned inline following the request. When you don't specify DUSTERR=INLINE, control is returned at the address specified in the ERRET operand of the NATTACH macroinstruction. The request is not honored.
	TU#DDTBE	NATTACH table not full-word aligned	
	TU#DERRA	Error return address not in user region	
	TU#DNAPP	APPS-name requested does not match any LOCAP name	
	TU#DCAPP	APPS-name requested is already attached to another CUP	
	TU#DNADU	Duplicate NATTACH from same user	
	TU#DRAPP	APPS-name requested is a remote LOCAP	
	TU#DGSHT	NATTACH while GUST shutdown in progress	
	TU#DGBL	Your program attempted an NATTACH to a global CCA that was not active.	
	TU#DPASS	Invalid password	

NDETACH

2.22.1.6. Detach Network (NDETACH)

Function:

This macroinstruction detaches your program from a global network. Following execution of this macro, ICAM will accept only the GAWAKE macro until your program issues another NATTACH macro.

This macro applies to global network systems only.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	NDETACH	network-name [,MF=({(D,C,N)[,{prefix-code,*}]},{E,{list,(1)}},{L})]

Positional Parameter 1:

network-name
Is the name of the global network.

MF Keyword Parameter:

MF=
Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

2.22.1.6.1. NDETACH Error Processing

When you specify DUSTERR=INLINE in the LOCAP macroinstruction and an error occurs during macroinstruction processing, control returns inline to the address following the NDETACH macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the NDETACH parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#NER1 error codes in byte 2.

Table 2-24 describes the error conditions that may be detected during the execution of this macroinstruction.

Table 2-24. NDETACH Error Conditions

Byte	Error Code	Cause/Condition	Result
TQ#NER1	TU#DDTAE	NDETACH table outside user region	When you specify DUSTERR=INLINE, control is returned inline following the request. When you don't specify DUSTERR=INLINE, control is returned at the address specified in the ERRET operand of the NATTACH ERRET=address. The request is not honored.
	TU#DDTBE	NDETACH table not full-word aligned	
	TU#DDTLE	NDETACH table length incorrect	
	TU#DFE	NDETACH table flags incorrect	

NOTES:

1. The DSECT for this table is TQ#DSCTS.
2. The TQ#NER2 error byte does not apply to NDETACH and always contains a zero.
3. Your program is canceled when:
   - you issue an NDETACH without a previous NATTACH; or
   - you issue an NDETACH after a previous NDETACH.

SESCON

2.22.1.7. Establish a Dynamic Session (SESCON)

Function:

This macroinstruction lets your program open, close, accept, reject, abort, or confirm the closing of a dynamic session with another end user. An end user is a terminal, a process file, or another program. In a public data network, this macroinstruction can, at your option, declare a dynamic session to be over a permanent virtual circuit.

You use this macroinstruction to create the SESCON executable code and related parameter list, or you can use it to create the executable code only or the parameter list only. You can create six different kinds of SESCON parameter lists depending on the type of function you want to perform. The six parameter lists are shown in Figure 2-16. The figure shows:

- the contents of each field;
- the operand in the SESCON macroinstruction you use to create the field;
- the label of each field from the DSECT; and
- whether ICAM or your program normally controls the contents of the field.

Table 2-25 shows the required and optional parameters your program supplies.

Table 2-26 shows the formats of the parameter lists.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	SESCON	[FUNCT={OPEN / CLOSE / ABORT / OPNACC / OPNREJ / CLSCNF}] ,MYID=four-char-id ,TONAME=end-user-name [,INQNAME=input-destination-name] [,PVC=name] [,MF={({D/C/N}[,{prefix-code / *}]) {(E,{list / (1)}) {L}

Label:

symbol
Is a 1- to 8-character label that identifies the executable code, parameter list generated by this macroinstruction, or its DSECT when the MF=D operand is specified.

FUNCT Keyword Parameter:

Specifies the function this macroinstruction performs. This operand is required except when you specify MF=D.

FUNCT=OPEN
This is a request to establish a session.

FUNCT=CLOSE
This is a request to close a session in progress in an orderly manner.

FUNCT=ABORT
This is a request to abort a session because your program detected an error.

FUNCT=OPNACC
This is a request to accept a session initiated by another end user.

FUNCT=OPNREJ
This is a request to reject a session initiated by another end user.

FUNCT=CLSCNF
This is a request to confirm the closing of a session initiated by another end user.

MYID Keyword Parameter:

MYID=four-char-id
MYID is a 4-character identifier your program assigns to a dynamic session. ICAM places this identifier in all *datagrams* it delivers to your program that relate to a particular *session*. Your program supplies MYID when it issues a TYPE=OPEN or TYPE=OPNACC SESCON macroinstruction.

TONAME Keyword Parameter:

TONAME=end-user-name
Specifies the name of the end user as stated during network definition. This parameter is required for FUNCT=OPEN only.

INQNAME Keyword Parameter:

INQNAME=input-destination-name
Destination queue for input messages coming from a terminal included in the session being opened or accepted.

Messages coming from a terminal are normally placed on a queue defined by the INPUT operand of that terminal's TERM macroinstruction. However, you can specify that messages for this session are to be placed on a different queue, such as the low priority queue of a process file, locap file, or terminal output queue. If you want the input traffic to go to the destination specified in the TERM macroinstruction, you can default this operand.

PVC Keyword Parameter:

PVC=name

An optional parameter for public data networks indicating that the dynamic session is through a permanent virtual circuit. The name is the symbolic name of a PVC macroinstruction in the network definition that defines a permanent virtual circuit connecting two end users.

MF Keyword Parameter:

MF=

Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

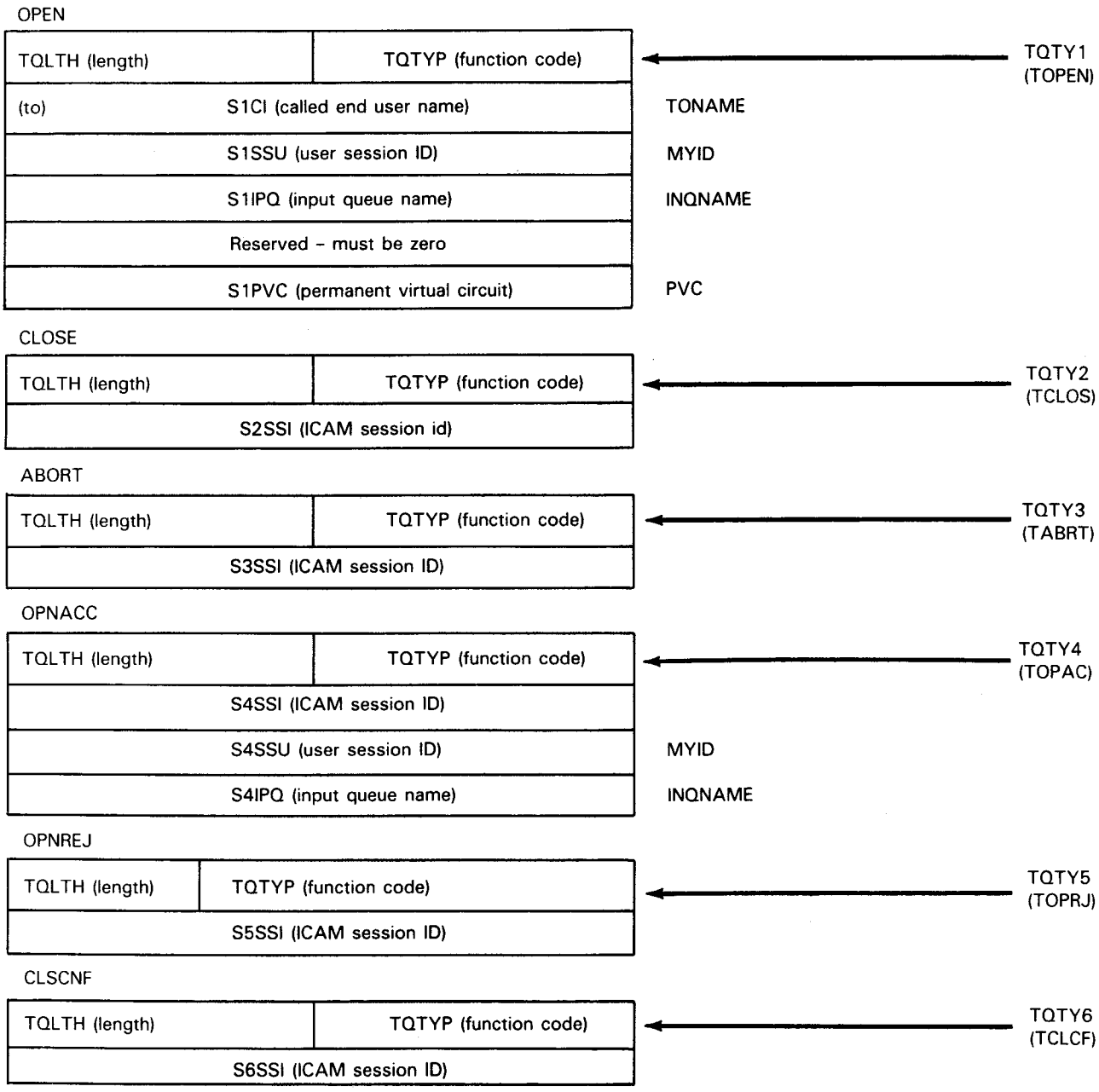
If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

Table 2-25. Required and Optional SESCON Parameters

FUNCT=	Parameters			
	User Session ID	TONAME	INQNAME	ICAM SESSION ID
OPEN	R	R	O	N/A
OPNACC	R	N/A	O	R
OPNREJ	N/A	N/A	N/A	R
CLOSE	N/A	N/A	N/A	R
CLSCNF	N/A	N/A	N/A	R
ABORT	N/A	N/A	N/A	R

LEGEND:

R - required
O - optional
N/A - not applicable



NOTE:

Each session control parameter list label is prefixed by the 1- to 3- character label you specify in your SESCON DSECT call. ICAM's session id is given to you in an open or open accept datagram. You must then place it in all subsequent SESCON parameter lists before issuing the actual macroinstruction service request (call).

Figure 2-16. Session Control (SESCON) Parameter List Format

Table 2-26. SESCON Parameter List Detailed Field Descriptions (Part 1 of 2)

Byte	Label*	Type and Length	Content	Word
0	TQLTH	H	Length of packet	1
2	RTN	X	Return function	
2	TQTYP	H	Function code	
3	(FUNC)	(X)	Function code	
	TQTY1 (TOPEN)		Open request	
	TQTY2 (TCLOS)		Close request	
	TQTY3 (TABRT)		Abort request	
	TQTY4 (TOPAC)		Open accept	
	TQTY5 (TOPRJ)		Open reject	
	TQTY6 (TCLCF)		Close confirm	
	TQTYA		DMI raise credit limit	
	TQTYB		DMI lower credit limit	
	TQTYC		Resiliency-port abort	
	TQTYD		Queuer-send AV	
For Open Request SESCON				
4	S1CI	F	To name-called end user	2
8	S2SSU	F	User's session ID	3
12	S1IPQ	F	Input queue name	4
16	S1PPR	H	Reserved	5
18	S1DEV	H		
20	S1PVC	F	PVC for public data networks	6
For Close Request SESCON				
4	S2SSI	F	ICAM's session ID	2
For Abort SESCON				
4	S3SSI	F	ICAM's session ID	2
8	S3RCD	H	Reason code	3
10		H	Unused	
For Open Accept SESCON				
4	S4SSI	F	ICAM's session ID	2

Table 2-26. SESCON Parameter List Detailed Field Descriptions (Part 2 of 2)

Byte	Label*	Type and Length	Content	Word
8	S4SSU	F	User's session ID	3
12	S4IPQ	F	Input queue name	4
For Open Reject SESCON				
4	S5SSI	F	ICAM's session ID	2
8	S5RCD	H	Reason code	3
		H	Unused	
For Close Confirm SESCON				
4	S6SSI	F	ICAM's session ID	2

* Labels are prefixed by default prefix I or by a 1- to 3-character prefix assigned by the MF parameter on the SESCON macroinstruction.

Programming Notes:

1. If you use the parameter list option MF=L and you do not use macroinstructions to specify the following fields, you must ensure that they are properly set in the parameter list before your program issues the SESCON call. (If you do not use the MFL option, the fields are properly set by the macroinstruction.) PNOTES may be generated to indicate that certain fields are required. The following describes the type of SESCON function, the related field to be defaulted, the parameter list field name, and the required value.

SESCON Function	SESCON Operand	Parameter List Field	Value Required
OPEN	INQNAME	xS1IPQ	X'40404040' (4 spaces)
OPNACC	INQNAME	xS4IPQ	X'40404040' (4 spaces)

where:

x
Is a 1-character user-supplied optional prefix.

2. ICAM session ID is a 4-character session identifier *established by ICAM.* It identifies a particular dynamic session. Your program receives the session identifier in a control datagram and it must save it (unchanged) and use it in any subsequent SESCON call that relates to that session – that is, when your program accepts, rejects, aborts, closes, or confirms a closed session. Your program moves the session identifier to the appropriate field in a SESCON parameter list prior to issuing the SESCON call. If your program initiates a session, the session identifier is delivered to your program with the *open accept* datagram. If your program is the one called, the session identifier is delivered in the *open* datagram. Figure 2-17 shows the organization of the different SESCON parameter lists and the labels used for the session identifier field in each.

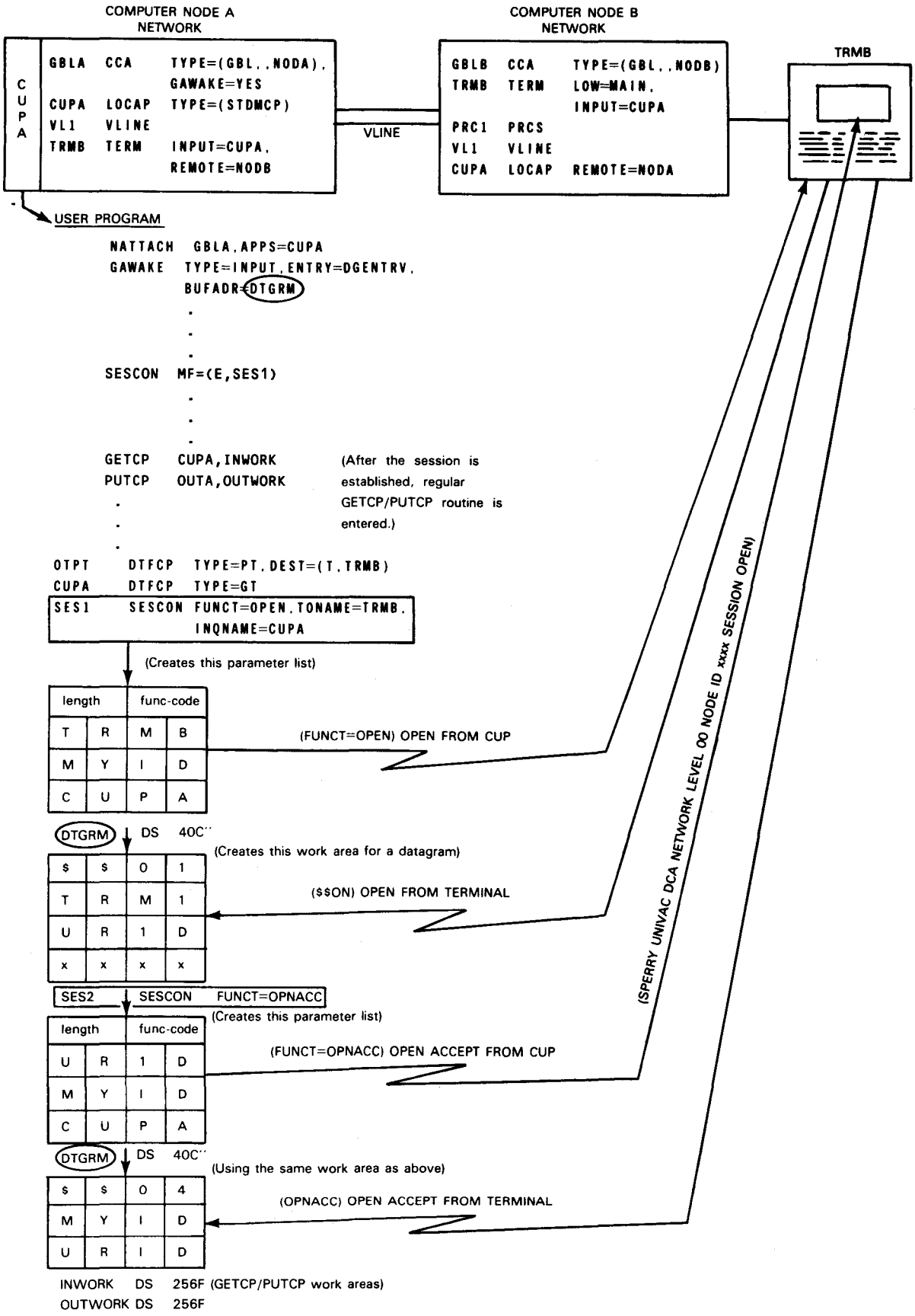


Figure 2-17. Pictorial Diagram of a Dynamic Session

If you want to establish multiple dynamic sessions, you must create your own session table to store user session and ICAM session identifier values.

Examples:

The following examples show various ways to write a SESCON macroinstruction. Notice the use of the MF=E operand to point to a SESCON parameter list in the nonexecutable portion of the program. The MF=E type is contained in the executable code. An MF=L type creates only the parameter list that is contained in the nonexecutable code.

```
SESA     SESCON  MF=(E,(1))                  Address of SESCON parameter list is in register 1
SESB     SESCON  MF=(E,SESCL)                Address of SESCON parameter list is label SESCL
SES1     SESCON  FUNCT=OPEN,TONAME=TRMA,     Creates an open parameter list
                 MYID=AB01,INQNAME=CUPA,MF=L
SES2     SESCON  FUNCT=OPEN,TONAME=TRMB,     Creates an open parameter list
                 MYID=AB02,INQNAME=CUPB,MF=L
SESCL    SESCON  FUNCT=CLOSE,MF=L            Creates a close SESCON parameter list
SESABRT  SESCON  FUNCT=ABORT,MF=L            Creates an abort SESCON parameter list
DSESCON  SESCON  MF=(D,SES)                  SESCON DSECT call with prefix to be SES
```

If you want, you can use only one SESCON macroinstruction and change the functions. For example, if your program receives a datagram in a buffer specified in GAWAKE as DATAGRAM and receives control at the entry you specified as DGENTRY, you can send a SESCON accept back to the calling terminal by using the labels shown in Table 2-26.

```
DGENTRY  STM 0,15,SAVEREG          Store entry register addresses
         USING DSDTGRM,1           Datagram cover register
         LA  1,DATAGRAM            Load datagram area address to map
         MVC EUTAG,DATK1CGI        Save end user name
         MVC SESSID,DATKISSI       Save ICAM session ID
         BAL 12,ACCREQ             Branch to accept request routine
          .
          .
          .
ACCREQ   ST  12,RETSVC             Store entry address
         USING DSESCON,1           SESCON cover register
         LA  1,SES1                Load SESCON address to map
         MVI SESTQTYP+1,SESTQTY4   Set function code to open accept
         MVI SESTQLTH+1,SESS4LT    Set length to four
         MVC SESS4SSU(4),MYID      Pick up session id and insert
         MVC SESS4SSI(4),SESSID    Pick up ICAM session ID and insert
         MVC SESS4IPQ(4),INQNAME   Insert input queue name
         SESCON  MF=(E,(1))        Execute the SESCON parameter list
                                   defined in the nonexecutable code
                                   (whose address is in register 1)
```

(cont)

```
            L    12,RETSVC                  Load return address
            BR   12                         Return
             .
             .
             .
            EOJ
SES1        SESCON  FUNCT=OPEN,MYID=AB01,   SESCON parameter list
                    TONAME=TRM1,INQNAME=CUPA,MF=L
DSESCON     SESCON  MF=(D,SES)              SESCON  DSECT with SES prefix
DSDTGRM     CONTDTG MF=(D,DAT)              Datagram DSECT with DAT prefix
DATAGRAM    DC      40C' '                  Define datagram buffer area for input
SESSID      DC      F'0'                    Define save area for ICAM session ID
EUTAG       DC      F'0'                    Define save area for end user name
MYID        DC      C'CP01'                 Define session id for my program (arbitrary)
```

This coding shows how you could retrieve information from a datagram and pass it on to a parameter list created by a SESCON macroinstruction. Using the datagram DSECT called DSDTGRM, the datagram work area in your program is mapped by doing a load address (LA) of the work area. The calling end user name and the session id provided by ICAM are saved. If the input datagram was determined to be an open, a branch is taken to the routine that changes the open SESCON to an open accept SESCON. In this routine, using a SESCON DSECT called DSESCON, the session control parameter list is mapped by doing a load address of the open SESCON macroinstruction parameter list. The function code and length are set to an open accept function. Your program session id and the ICAM session id are picked up and placed in the session control parameter list.

If you wanted to change the input destination from the destination named on the INPUT= operand of the TERM macroinstruction in the network definition, you use the INQNAME operand on the SESCON macroinstruction, or you could set it here in your user code. Care must be exercised, however, that all default parameters must be set before issuing the SESCON service call. For instance, if the INQNAME parameter is not used here because you are using the INPUT parameter destination of the TERM macroinstruction, and you are *not* using a SESCON macroinstruction INQNAME parameter, then you must set the SESS4IPQ field to zeros within your user program code.

Once all your parameters are set, you can issue the SESCON. In the coding shown, an MF=E form is used to create an imperative macroinstruction within the executable code that points to a parameter list SES1 in the nonexecutable code. An MF=L form of the macroinstruction is used to create the SES1 parameter list in the nonexecutable code. This method is most useful for multiple dynamic session SESCON service requests.

See Figure 2-40 for a working example of a dynamic session program.

2.22.1.7.1. SESCON Error Processing

When your program issues a SESCON call, ICAM validates the request and returns control to your program inline. Your program should then determine whether its request was valid and whether ICAM accepted it for further processing. Your program does this by examining byte 3 of register 0. If byte 3 is zero, you can assume that the request was accepted by ICAM for further processing. If it is not zero, the request was rejected for the reasons specified in Table 2-27.

Table 2-27. SESCON Macroinstruction Error Return Conditions

Error Value	Meaning
xER2	Parameter list supplied is not word aligned.
xER3	User program has not issued GAWAKE macroinstruction to receive datagram response.
xER4	Parameter list supplied is not in user program region.
xER5	Function code specified is not valid.
xER6	ICAM session identifier supplied cannot be identified by ICAM. (This may be because the session is already aborted, or the name of the input queue (INQNAME) cannot be located in the current network.)
xER7	Session is currently being aborted.

NOTES:

1. x is an optional user-supplied prefix.
2. For the SESCON macroinstruction, ICAM always returns errors inline, regardless of whether you specify DUSTERR=INLINE in the LOCAP macroinstruction. (See 2.12 for details.)

TRMREL

2.22.1.8. Disconnect a Terminal from Circuit-Switched DATEX-L (TRMREL)

Function:

Disconnects a terminal from a circuit-switched DATEX-L public data network. You can use this instead of specifying the DISIN/DISOUT operands of the TERM macroinstruction in your network definition.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	TRMREL	terminal-name [,MF=(L | D | (E [parameter-list-address | (1)]))]

Label:

symbol
A 1- to 4-character label identifying this macroinstruction.

Positional parameter 1:

terminal-name
The network definition symbolic name of the terminal (TERM) to be disconnected.

MF=
Identifies a parameter list whose address may be explicitly specified or implicitly passed to the called routine through register 1. See S-type macroinstructions described in the ICAM concepts and facilities, UP-8194 (current version).

If this parameter is omitted, both a parameter list required by the called ICAM routine and the linkage to it are generated.

2.22.1.8.1. TRMREL Error Processing

ICAM returns control inline to the address following the TRMREL macroinstruction. After macroinstruction processing, register 1 contains the address of the TRMREL parameter table. Register 0 contains one of the following codes:

0 = circuit disconnected successfully

4 = invalid terminal name

8 = register 1 contents invalid (address of user parameter list)

NOTE:

For the TRMREL macroinstruction, ICAM always returns errors inline, regardless of whether you specify DUSTERR=INLINE in your program's CCA and LOCAP macroinstructions. (See 2.12 for details.)

2.22.2. Relinquishing and Acquiring Communications Control

The CYIELD macroinstruction is issued by your program to relinquish control following a series of GETCP/PUTCP macroinstructions that had the IRL set. CYIELD is not necessary if a GETCP or PUTCP is issued without IRL set.

Two other important uses of CYIELD are:

- Yielding control to ICAM in order to receive a datagram.
- Asynchronous completion of an event, such as line error status and unsolicited datagrams. Your program should use CYIELD to receive what was originally expected. For instance, if your program issues a PUTCP and expects a completion, but receives an unsolicited datagram instead, it should process the datagram and then issue a CYIELD to receive the PUTCP completion.

The CYIELD macroinstruction passes control to ICAM activity via the OS/3 supervisor. If an activity is outstanding for that task, your program is immediately awakened (scheduled) and given control via the OS/3 supervisor. If no activity is outstanding, the task is suspended. You may activate (awaken) the communication task at any time by executing the CAWAKE macroinstruction from any other task of that job. The occurrence of an activity within ICAM for the communications task activates the task, provided that it is in an idle (CYIELD) condition.

For efficiency, direct all noncommunications I/O to a task other than a communications task. The communications activity directed to the communications task by ICAM can continue to flow while the printer/reader disk I/O is functioning. The CAWAKE macro synchronizes the activities of the communications and noncommunications tasks.

By using the GAWAKE macroinstruction, your program can awake any other program within the system if the programs are known to each other and the program to be awakened is a communication user registered with ICAM. This facility permits noncommunication users to call a communication user program with a set of parameters in the form of a datagram. The parameters are contained in the work area of the sending program and are transferred to the work area of the receiving program without change.

The awake function is initiated when a task executes the GAWAKE macroinstruction. A register (R1) has the address of the user's work area that contains the following types of information:

- Name of the task to be awakened
- Parameter buffer address
- Parameter buffer length

The initiating task is automatically returned control with a status that indicates one of the following conditions:

- Accepted
- Rejected; no facilities available
- Rejected; receiver's name not registered

ICAM moves the user's parameters into one or more network data buffers and queues them into the facility table associated with the named receiving task. No delivery notices are ever returned to the sending task; there is no guarantee that the datagram is delivered.

If the named receiving task is idle or when it becomes idle (in a CYIELD condition), ICAM transfers the message into the user's work area and awakes that task at a previously specified entry address.

CYIELD

2.22.2.1. Release of Control (CYIELD)

Function:

Releases control of your program to ICAM while waiting for a communications I/O completion or an ICAM event. You issue the CYIELD macroinstruction in your program following a series of GETCP/PUTCP macroinstructions with IRL set or when you want your program to be dormant until an ICAM event occurs. Such an event could be another program sending you a datagram, or ICAM sending you a control datagram as a result of a terminal user or another program signing on for a dynamic session. CYIELD is not necessary if you issue a GETCP or PUTCP without IRL set.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	CYIELD	

2.22.2.1.1. CYIELD Error Processing

ICAM always returns control inline following the CYIELD macroinstruction. No return codes are supplied. Only one type of error can occur during macroinstruction processing – your program is canceled with error code 450 when the task issuing the CYIELD is not an active communications task.

CAWAKE

2.22.2.2. Awake Communications Task (CAWAKE)

Function:

Activates a communications program in a CYIELD condition. If your program is operating in a multitask environment, any noncommunications task or island code can activate a communications task by issuing CAWAKE. This gives control to the communications task at the instruction immediately following CYIELD.

You can use either the CAWAKE or GAWAKE macroinstruction to activate a communications program, but the two instructions have these important differences:

1. CAWAKE can only activate another communications program within the same job; GAWAKE can be issued from any program (even a noncommunications program) and activate any communications program in any job.

2. Your program can issue datagrams using GAWAKE, but not with CAWAKE.

3. When you issue CAWAKE, control is given to the activated program at the instruction following CYIELD. When you issue GAWAKE with the TYPE=OUTPUT operand, control is given to the activated program at the entry address designated when GAWAKE was issued with the TYPE=INPUT operand.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	CAWAKE	

2.22.2.2.1. CAWAKE Error Processing

ICAM always returns control inline following the CAWAKE macroinstruction. No return codes are supplied. Only one type of error can occur during macroinstruction processing - your program is canceled with error code 450 when the program issuing the CAWAKE has no active communications task.

GAWAKE

2.22.2.3. Awake Global Task (GAWAKE)

Function:

Enables a noncommunication program or another communications program to awaken a communications program and pass datagrams to it. You make your communications program eligible to be activated by another program and receive datagrams from it by having your program issue the GAWAKE macroinstruction to itself using the TYPE=INPUT operand. In addition, you must specify GAWAKE=YES in the CCA network definition macroinstruction and provide network buffers during system generation to hold the datagrams.

To pass datagrams to a communications program, the receiving communications program must first be eligible to receive them as described previously. Then, you must specify GAWAKE in your sending program using the TYPE=OUTPUT operand.

You can use either the CAWAKE or GAWAKE macroinstruction to activate a communications program, but the two instructions have these important differences:

1. CAWAKE can only activate a communications program within the same job; GAWAKE can be issued from any program (even a noncommunications program) and activate any communications program in any job.

2. Your program can issue datagrams using GAWAKE, but not with CAWAKE.

3. When you issue CAWAKE, control is given to the activated program at the instruction following CYIELD. When you issue GAWAKE with the TYPEOUTPUT operand, control is given to the activated program at the entry address designated when GAWAKE was issued with the TYPE=INPUT operand.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GAWAKE	TYPE={INPUT / OUTPUT} [,ENTRY=label] [,APPS=name] [,BUFADR=label] [,BUFLTH=n] [,MF={L / (E,packet.address)}]

Parameters:

TYPE=

Defines the packet being generated as the recipient of a datagram or the originator of the datagram. TYPE=INPUT indicates that the generated code is an input packet and is used by your program to register itself as an eligible GAWAKE recipient. When TYPE=INPUT is specified, the ENTRY=label operand must also be defined.

TYPE=OUTPUT indicates that the generated code is an output packet and is used to activate a communications program and optionally pass a datagram.

If this operand is omitted and the MF operand is also omitted, the packet generated defaults to output.

ENTRY=label

Specifies the address within your program to receive control when your program is activated by another program. This operand is required when TYPE=INPUT is specified but is not valid for TYPE=OUTPUT.

APPS=name

Specifies the registered LOCAP name of the recipient program or, for dedicated networks, the label of the network name.

BUFADR=label

Specifies the address of the storage area within a receiving or sending program for a datagram. When TYPE=OUTPUT is specified, BUFADR=label defines where the data to be transferred is stored in the sending program. When TYPE=INPUT is specified, BUFADR=label specifies where data received by a registered communications task is to be stored. This address must be on a full-word boundary.

If BUFADR is omitted, it indicates that no datagram is to be transferred. If dynamic session establishment is used, BUFADR must be specified.

BUFLTH=n

Specifies the length of the storage area used to contain a datagram. It must be specified as a decimal number from 1 to 32,767. If this operand is not specified, code is generated so that no data is transferred. If dynamic session establishment is used, n should be at least 40 bytes, which is the minimum size for control datagrams.

MF=

Specifies how this macro is to be generated. If the MF operand is omitted, a packet of parameters and the macro are generated inline each time the GAWAKE macro is issued.

MF=L specifies that only the packet of parameters is to be generated. The packet can then be referenced by a GAWAKE macro having the MF operand specified as MF=(E,packet-address).

MF=(E,packet-address) specifies that only the SVC is generated with its subfunction code. Packet-address is the label assigned to the packet generated via the MF=L operand.

2.22.2.3.1. GAWAKE Error Processing

ICAM returns control inline to the address following the GAWAKE macroinstruction. After macroinstruction processing, register 1 contains the address of the GAWAKE parameter table. When byte 3 of register 0 contains a zero, the processing was successful. When it contains any of the codes shown in Table 2-28, an error has occurred.

Table 2-28. GAWAKE Error Conditions

Error Code	Condition/Cause
TN#ZER1	Parameter list not word aligned
TN#ZER2	Address of parameter list out of range
TN#ZER3	No length specified for given buffer
TN#ZER4	Given buffer is out of range
TN#ZER5	Recipient not ready or NATTACH must be done before GAWAKE
TN#ZER6	Recipient does not accept datagrams
TN#ZER7	Recipient accepts only register values
TN#ZER8	ICAM not configured for datagrams
TN#ZER9	ICAM resources not available
TN#ZER10	Open
TN#ZER11	ICAM resources not available
TN#ZER22	Entry address not aligned
TN#ZER23	Entry address out of range
TN#ZER24	Input length not specified
TN#ZER25	Input buffer not in range
TN#ZER26	NATTACH not done (input)

NOTES:

1. For the GAWAKE macroinstruction, ICAM always returns errors inline, regardless of whether you specify DUSTERR=INLINE in your program's CCA and LOCAP macroinstructions. (See 2.12 for details.)

2. ICAM returns these error codes only in byte 3 of register 0.

3. The equates for these error codes are defined in the DSECT for the GAWAKE parameter list. (See Table 2-29.) You can obtain the DSECT by calling TN#DSECT GAWAKE. See 2.17 for a discussion on obtaining DSECTs.

Table 2-29. GAWAKE Parameter List Detailed Field Descriptions

	Label	Type and Length	Content
Output*	TN#KOFLG	XL1	Error flags and indicators; see Table 2-28.
	TN#KOOP		Output packet indicator
	TN#KORGS		Send no data
	TN#KORBU		Send data in buffers
	TN#KOP1	XL1	Open
	TN#KOBLT	H	Length of output buffer
	TN#KOP2	XL1	Open
	TN#KOBAD	AL3	Address of output buffer
	TN#KOBAU		Full-word address of output buffer
	TN#KORLN	F	Recipient's LOCAP name (APPS)
	TN#KOSIZ		Size of output packet
Input**	TN#KIFLG	XL1	Error flags and indicators; see Table 2-28.
	TN#KIIP		Input packet indicator
	TN#KIRGS		Receives no data
	TN#KIRBU		Receives data in buffer
	TN#KGAWK	AL3	GAWAKE entry address
	TN#KGWKE		Full-word entry address
	TN#KIANB	H	Number of bytes input
	TN#KIBLT	H	Length of input buffer

* DSECT label is TN#KGAWO.
** DSECT label is TN#KGAWI.

2.22.3. Sending and Receiving Messages

There are two macroinstructions in this group: GETCP and PUTCP. GETCP transfers messages from ICAM to your program; PUTCP transfers messages from your program to ICAM.

GETCP

2.22.3.1. Access Queued Messages (GETCP)

Function:

Accesses messages from ICAM and identifies the area in your program that receives the message unit.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	GETCP	{filename / (1)} {workarea-address / (Ø)}

Parameters:

`filename`
Identifies the input DTFCP to be accessed.

`(1)`
Indicates the address of the file to be accessed is contained in register 1.

`workarea-address`
Identifies the area in your program that receives the message unit.

`(Ø)`
Indicates the address of the work area is contained in register 0.

NOTE:

Remember that this is the address of the work area prefix that specifies the length of the message unit that follows. The work area prefix must be on a half-word boundary.

Example 1:

The following examples show various ways to write a GETCP macroinstruction.

```
1       10    16
GET1    GETCP PRC1,INAREA1
GET2    GETCP (1),(Ø)
```

Example 2:

The following example shows how a GETCP macroinstruction is used within a typical program structure.

```
1         10    16
-------------------------------------------------------------------

CUP1      START Ø
          TM#DSECT  PRCS                  INPUT DTF PROC CALL
          BALR  1Ø,Ø                      Program initialization
          USING *,Ø                       Program initialization
          USING TM#PRCS,2                 INPUT DTF COVER REGISTER
           .
           .
           .
          NATTACH   NET1,APPS=INPT        Attach to a global network
           . (BAL code to initialize the DTFCP
           . before issuing the GETCP.)
           .
          GETCP  INPT,WORKIN              Issue the GETCP
           .
           .
           .
NOGET     LA    2,INPT                    Load the input DTFCP address
          CLI   TM#PERR,TM#PEIN           Error processing routine
          BE                              for a PUTCP. The
          CLI   TM#PERR,etc.              entry address is
          BE                              specified in the DTFCP.
          CLI   TM#PERR+1,etc.
          BE
           .
           .
           .
          EOJ
INPT      DTFCP TYPE=GT,ERRET=NOGET       Define the input file
WORKIN    DC    256 CL'△'                 Create a work area buffer
           .                              for input messages
           .
           .
          END
```

2.22.3.1.1. GETCP Message Transfer Units

At your option, GET processing transfers message data to your work area in one of three message units: complete messages, message segments, or single lines of the message.

Whether the transfer unit is a complete message or message segment depends upon the size of the user work area. GETCP processing transfers data to the user work area until all data in the message is transferred or until the work area is filled. When the work area is filled before all of the message text is transferred, control is returned to your program with a more data status indication in the DTFCP. One or more GETCPs must be executed to acquire the remaining message data.

The line transfer unit is selected when a flag, TM#PLTR, is set in byte TM#PDTF of the DTFCP. In this case, GETCP processing scans the message until an end-of-line DICE sequence is found. The text data in that line, excluding the DICE sequence, is then transferred to the user work area. Each GETCP executed results in the transfer of one line of the message. The user work area should be large enough to contain the longest line of the message.

NOTE:

The end-of-line (EOL) DICE sequences are stripped from a message only when the transfer unit is a single line of a message.

2.22.3.1.2. GETCP Message Restart

When operating with segment or line message units, your program can restart transfer of a message from the beginning of the message. To accomplish this, you set flag TM#PSHD in DTFCP byte TM#PSEG, and execute a GETCP macroinstruction. ICAM responds by retransferring the first segment or line of the message to the user work area.

2.22.3.1.3. GETCP Message Cancel

With segment or line message units, you can cancel a partially transferred message. If, before transfer of the last segment or line, you execute a GETCP macroinstruction with flag TM#PNHD set in the DTFCP byte TM#PSEG, ICAM dequeues and releases any buffers that contain data of the current message and transfers the next message (or first segment or line of the next message) to the user work area.

When there is no 'next' message available, the Get processing response is determined by the values of the IRL flag (TM#PIRL) and the no-message-available address (TM#PNMA).

2.22.3.1.4. GETCP Queueing Priority

The GETCP queueing structure is similar to the PUTCP destination queueing structure. A GETCP process file or LOCAP may also have up to three queues designated as having high, medium, or low priority. A specific queue may be accessed by setting a value of TM#EHIG, TM#EMED, or TM#ELOW in byte TM#PDTF of the DTFCP file table.

A fourth priority value, TM#EAVL, causes all three queues to be scanned, from high to low priority, for the first 'available' message. If a GETCP is executed with no priority value in byte TM#PDTF, a priority of 'available' is used as a default.

2.22.3.1.5. GETCP Error Processing

GETCP processing returns error indications for your program in two bytes of the DTFCP file table. The bytes are labeled TM#PERR and TM#PERR+1. As previously described, flag TM#PIER is also set in the status byte TM#PIND whenever an error flag is set in one of the two TM#PERR bytes.

The error indicators that may be returned due to GETCP operations are described in Table 2-17 (located with the DTFCP macroinstruction for Get functions). It is suggested that you read each of the items in the table carefully so your program can test for these possible errors and take appropriate action.

NOTE:

For the GETCP macroinstruction, ICAM always returns control to the specified address in the DTFCP macroinstruction ERRET= or NOMAV= keywords, regardless of whether you specify DUSTERR=INLINE in your program's CCA and LOCAP macroinstructions. If you don't specify an ERRET= address, control is returned inline following the macroinstruction. (See 2.12 for details.)

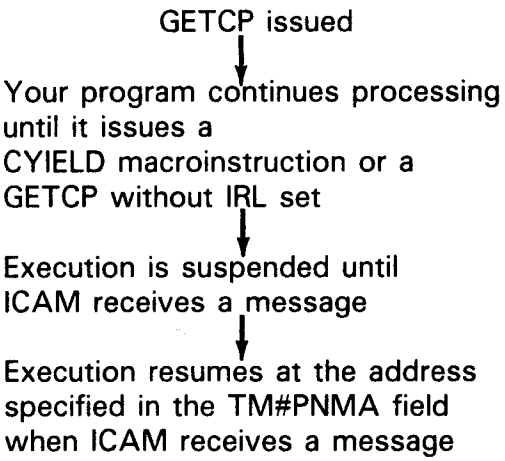
2.22.3.1.6. GETCP Status Indications

Two bytes of the DTFCP file table, TM#PIND and TM#PIND+1, are used by ICAM to return Get processing status indications to your program. The Get processing indicators that are returned to Get processing are described in Table 2-17 (located with the GETCP macroinstruction for Get functions). It is suggested that you read the table thoroughly so your program can test for these status indicators following each GETCP and take appropriate action.

2.22.3.1.7. Deferred Gets

ICAM gives you the option of suspending the execution of your program with the deferred Get facility when there are no input messages available to process. Execution of your program resumes when ICAM receives an input message.

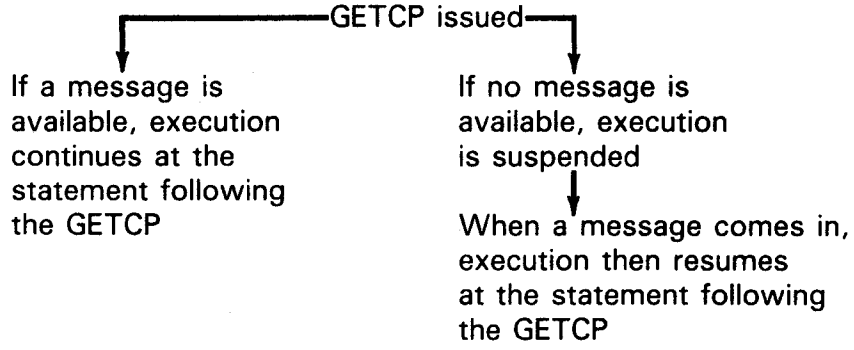
You use deferred Gets by means of the IRL setting and the CYIELD macroinstruction. Your program sets the TM#PIND field of the input DTFCP with TM#PIRL set. It then places the address where execution is to continue when a message is available in the TM#PNMA field of the input DTFCP. Once these steps are completed, the following events occur:



Because execution continues until your program issues CYIELD or a GETCP without IRL set, it can issue a number of deferred Gets before suspending execution. Then, when a message comes in, your program processes it, reissues the deferred Get, and issues CYIELD. This way your program becomes interrrupt-driven, executing only when there is a message to process.

If a message is available when the CYIELD is issued, execution immediately transfers to the address in TM#PNMA.

You can use the deferred Get without using IRL and CYIELD. Normally, when IRL is not set, the TM#PNMA field contains the address where control is to transfer if no message is available. If, instead of an address, TM#PNMA contains blanks, the following events occur:



With this method, only one deferred Get can be issued at a time. If you issue two deferred Gets without issuing a CYIELD macroinstruction, the work area prefix is overwritten, preventing the successful processing of the first GETCP.

If you are using a dedicated network and a line goes down, your program receives control at the address you specified in the ERRET operand of the NETREQ macroinstruction. If you want to continue processing, you must deal with the line-down condition by issuing a LNEREL followed by a LNEREQ for the line in question. If a deferred GETCP was outstanding when the line went down, it will still be outstanding when the line connection is reestablished. Therefore, the next time you want to issue a deferred GETCP in your program, you should issue a CYIELD instead. If you alter the associated DTFCP or the work area prefix between issuing the GETCP and the CYIELD, the results are unpredictable.

PUTCP

2.22.3.2. Transfer Messages (PUTCP)

Function:

Transfers messages from a program work area to ICAM.

Format:

LABEL	ΔOPERATIONΔ	OPERAND
[symbol]	PUTCP	{filename / (1)},{workarea-address / (Ø)}

Parameters:

`filename`
Identifies the output DTFCP file table that controls the message.

`(1)`
Indicates the address of the DTFCP file table is contained in register 1.

`workarea-address`
Identifies the area in your program that contains the message unit to be transferred.

`(Ø)`
Indicates the address of the user work area is contained in register 0.

NOTE:

Remember that this is the address of the work area prefix, which specifies the length of the message that follows. The work area prefix must be on a half-word boundary.

Example 1:

The following examples show various ways to write a PUTCP macroinstruction.

```
1       10    16
PUT1    PUTCP PUTF,WORKAREA
PUT2    PUTCP OUT1,WKAR3
PUT3    PUTCP MAIL,RECORD1
PUT4    PUTCP (1),(Ø)
```

Example 2:

The following example shows how a PUTCP macroinstruction is used within a typical user program structure.

```
1         10    16
CUP1      START Ø
          TM#DSECT  DEST              OUTPUT DTF PROC CALL
          BALR  1Ø,Ø                  Program initialization
          USING *,1Ø
          USING TM#DEST,3             OUTPUT DTF COVER REGISTER
           .
           .
           .
          NATTACH  NET1               Attach to a global network
           . (BAL code to initialize DTFCP and construct
           . messages in work area before issuing PUTCP.)
           .
          PUTCP OTPT,WORKOUT          Issue the put
           .
           .
           .
NOPUT     LA    3,OTPT                Load the output DTFCP address.
          CLI   TM#DERR,TM#DEIN       Error processing routine for a PUTCP
          BE
          CLI   TM#DERR,etc.          The entry address
          BE                          is specified in the DTFCP.
          CLI   TM#DERR+1,etc.
          BE
           .
           .
           .
          EOJ
OTPT      DTFCP TYPE=PT,ERRET=NOPUT   Define the output file
WORKOUT   DC    256 CL'△'             Create a work area buffer
          END                         for output messages.
```

NOTE:

You can temporarily store messages in a process file and later release them for transfer to their final destination by using the QTRANS macro. See the QTRANS macroinstruction, 2.22.4.6, for a description.

2.22.3.2.1. PUTCP Message Transfer Units

Messages may be transferred from your user work area to ICAM in three message units: complete messages, message segments, or single lines of a message. Two flags, TM#DHDR and TM#DTND in byte TM#DSEG, plus flag TM#DLTR in byte TM#DDTF are provided for you to select the message transfer unit. Table 2-30 shows the appropriate settings for the flags in relation to the message unit to be transferred.

Table 2-30. Message Unit Selection Flags

Message Unit	Byte TM#DSEG		Byte TM#DDTF
	Flag TM#DHDR	Flag TM#DTND	Flag TM#DLTR
Put Complete Message	Set	Set	Reset
Put First Segment	Set	Reset	Reset
Put Intermediate Segments	Reset	Reset	Reset
Put Last Segment	Reset	Set	Reset
Put First Line	Set	Reset	Set
Put Intermediate Line	Reset	Reset	Set
Put Last Line	Reset	Set	Set/Reset

A complete message is transferred from the user work area to network buffers by execution of a single PUTCP macroinstruction. The message length is determined by the text byte count contained in the message work area prefix. The text byte count may not exceed $2^{15}-1$ (32,767) bytes.

A message segment is simply a portion of a complete message. Transfer of a segment implies that multiple PUTCPs are executed to transfer the complete message. Segment size is determined by the count in the user work area prefix and may vary from segment to segment.

When the selected message transfer unit is 'line', ICAM appends an end-of-line DICE sequence after the last text byte transferred from the user work area for each PUTCP executed, except for the last line of the message. Lines may also vary in size, but your work area should be large enough to accommodate the longest line of the message.

NOTE:

When a PUTCP is successfully processed, byte TM#DSEG is cleared prior to the return of control to your program. TM#DSEG is not cleared if the return of control is due to an error. Flag TM#DLTR in byte TM#DDTF is never reset.

2.22.3.2.2. PUTCP Message Abort

Transmission of a message is not initiated until the whole message has been successfully transferred to ICAM and queued. When the message transfer units of segment or line are selected, a partially queued message can be purged from the system at any time prior to the transfer of the last segment or line. This is performed by setting flag TM#DHDR in byte TM#DSEG and executing a PUTCP.

2.22.3.2.3. PUTCP Queueing Priority

Any destination may be associated with one, two, or three queues, with the queues designated as being of high, medium, or low priority. The destination queues associated are determined at network generation time. When your program is assembled, the DTFCP file tables are generated with the user-selected or default priority value in byte TM#DDTF. During user program execution, you may select the queueing priority for a given message by changing the value in byte TM#DDTF, TM#EHIG, TM#EMED, or TM#ELOW, as desired. The priority value must be changed prior to execution of the first PUTCP for the message.

In addition to high, medium, or low priorities, a message may be assigned a top priority by placing the value TM#ETOP into byte TM#DDTF. In this case, the message is queued as the first message on the high priority queue, and is the next message transmitted to the designation.

NOTE:

Disk queues do not support top priority.

2.22.3.2.4. PUTCP to Auxiliary Devices

Output messages to auxiliary devices may be either control messages or text messages. To transmit a message for an auxiliary device, the device identification may be placed in byte TM#DDVC of the DTFCP file table and the command/function code placed in byte TM#DSPEC. In addition, flag TM#DAFN must be set in byte TM#DSEG. When the auxiliary device message does not require the transfer of text, the work area prefix must contain a byte count of zero.

2.22.3.2.5. PUTCP Error Processing

Two bytes of the DTFCP file table, TM#DERR and TM#DERR+1, are used to return PUTCP processing error indications for the user program. The flags in the two TM#DERR bytes indicate the type of error detected; however, flag TM#DIER in byte TM#DIND indicates an error status. TM#DIER is always set in addition to a TM#DERR byte flag.

Error conditions that may result from PUTCP operations are listed in Table 2-13 (located with the DTFCP macroinstruction for Put functions). It is suggested that you read each of the items in the table so your program can test for these possible errors and take proper action.

NOTE:

For the PUTCP macroinstruction, ICAM always returns control to the specified address in the DTFCP macroinstruction ERRET= or NOMAV= keywords, regardless of whether you specify DUSTERR=INLINE in your program's CCA and LOCAP macroinstructions. If you don't specify an ERRET= address, control is returned inline following the macroinstruction. (See 2.12 for details.)

2.22.3.2.6. PUTCP Status Conditions

Byte TM#DIND in the DTFCP file table returns Put processing status indications to your program. The indicators that may be returned are described in Table 2-13 (located with the PUTCP macroinstruction for Put functions).

2.22.4. Displaying and Altering Network Status

The macroinstructions composing this group are used to:

- suspend and resume the transmission of messages from queues;
- interrogate network status;
- interrogate and alter terminal specifications;
- interrogate and alter polling sequences; and
- interrogate communications control area tables.

The display/alter macroinstructions are:

- CCACPY

  Copies selected network information.

- QCLEAR

  Allows the clearing of queues without transmission.

- QDEPTH

  Determines the number of messages on a queue.

- QHOLD

  Suspends transmission of output messages from a queue or queues.

- QRELSE

  Allows transmission from a previously held queue or queues.

- QTRANS

  Transfers messages from a queue of one process file or terminal to another.

- RELEASM

  Allows transmission to a terminal, held on intercept, to resume.

- TRMREP

  Changes a phone number in the line control table.

CCACPY

2.22.4.1. Copy Selected Network Information (CCACPY)

Function:

Retrieves line name, logical line number, terminal index, line terminal count, and device control tables from the communications control area and stores the information where specified in your program. You can use this macroinstruction to verify terminal names for user program traffic and find invalid names before the first GETCP/PUTCP is issued.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	CCACPY	$\left\{\begin{matrix}\text{workarea}_1\text{-address},\text{length}_1,\text{workarea}_2\text{-address},\text{length}_2\\(1)\end{matrix}\right\}$

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2–18 and Table 2–31.

Parameters:

$\text{workarea}_1\text{-address}$
Identifies the address in your program where the names of active terminals are stored. The end of the terminal name list is marked by a word of hexadecimal F(FFFFFFFF_{16}). The format of the terminal name list that you construct in your program is shown in Figure 2–19 and Table 2–32.

(1)
Indicates you have preloaded the address of your own parameter list in register 1. This parameter list must be on a full-word boundary and conform to the format in Figure 2–18 and Table 2–31.

length_1
Indicates the number of bytes in workarea_1.

$\text{workarea}_2\text{-address}$
Identifies the address in your program where the communications control area information is to be stored. The format of the CCA information table that you

construct in your program is shown in Figure 2-20 and Table 2-33. The end of the information table is marked by a word of hexadecimal Fs.

$length_2$
Indicates the number of bytes in $workarea_2$.

Example 1:

```
1         10    16
RDCCA1    CCACPY TERMLIST,LTRMLST,CCAINF,LCCAINF
RDCCA2    CCACPY TLIST,LTLIST,CCADATA,LCCAD
```

BYTE	0	1	2	3	WORD
0	error-half word		unused	function-code	1
4	CCA-information-table-address				2
8	terminal-name-list-address				3
12	CCA-information-table-length		terminal-name-list-length		4

Figure 2-18. CCACPY Parameter List Functional Field Description

Table 2-31. CCACPY Parameter List Detailed Field Description

Byte	Field		Content	Comment	Word
	Label*	Type and Length			
0	TQ#QERR	H	Error half word	Bits set by ICAM	1
2		XL1	Unused		
3	TQ#QFUNC	XL1	Function code indicator	Identifies CCACPY function	
4	TQ#QCAI	F	Address of user area holding CCA information table	CCA information stored by ICAM into this user work area	2
8	TQ#QTNT	F	Address of user area containing terminal name list	Terminal names obtained by ICAM from this user work area	3
12	TQ#QLEN1	XL2	Number of bytes in CCA information table		4
14	TQ#QLEN2	XL2	Number of bytes in terminal name list		

* The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

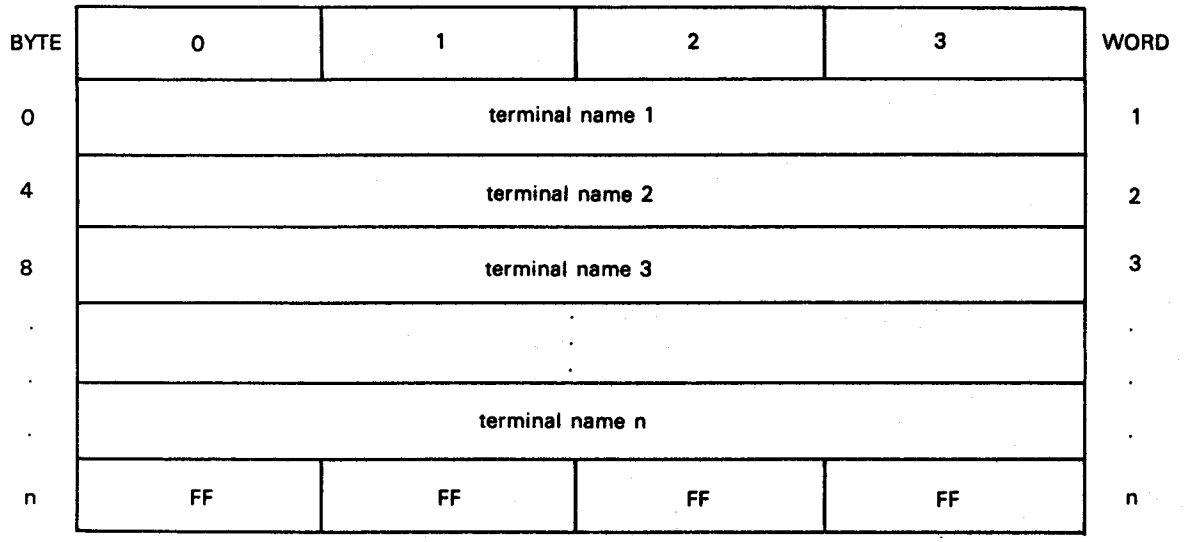


Figure 2-19. CCACPY Terminal Name List Functional Field Description

Table 2-32. CCACPY Terminal Name List Detailed Field Description

Byte	Field		Content	Comment	Word
	Label*	Type and Length			
0	TQ#CTRMI	CL4	Terminal name	Table consists of a series of these items.	1

* The DSECT for the table is TQ#CCINP, which is part of TU#DSTZ.

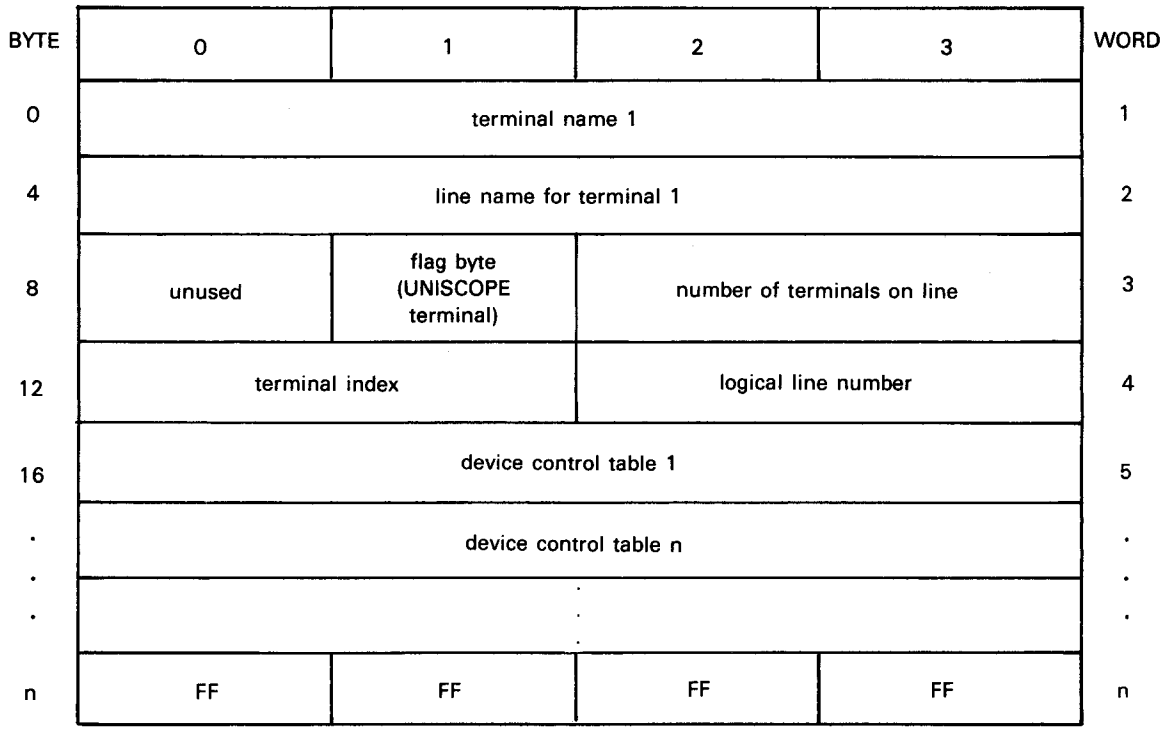


Figure 2-20. CCA Information Table Functional Field Description

Table 2-33. CCA Information Table Detailed Field Description

Byte	Field		Content	Comment	Word
	Label*	Type and Length			
0	TC#CTRM	CL4	Terminal name		1
4	TC#CLINE	CL4	Line name		2
8	TC#CRESV	XL1	Unused		3
	TC#CU100	XL1	Flag indicator	Indicates screen type (64/80 column)	
	TC#CNRT	H	Number of terminals on line		
12	TC#CTIDX	H	Terminal index		4
	TC#CLLN	H	Logical line number		
16	TC#CDCT	F	Device control table name	All DCTs linked to the TCT are included as part of the table entry.	5

* The DSECT for this table is TC#CCOTP, which is part of TU#DSTZ.
TC#CU80C 80-column UNISCOPE terminal or a non-UNISCOPE terminal
TC#CU64C 64-column UNISCOPE terminal

Example 2:

This example uses the first format to show how a CCACPY macroinstruction could be coded in a user program. Using this format, the CCACPY macroinstruction expansion generates a CCACPY parameter list inline. The work area addresses and lengths specified in the macroinstruction are placed in the parameter list. In this particular example, three CCACPY macroinstructions are shown, each creating inline its own parameter list pointing to its respective terminal table (TRMLST1, TRMLST2, TRMLST3) and information table (CCAINF1, CCAINF2, CCAINF3). You must build the terminal name list and information table list within your user program, using the labels you supplied as parameters in the macroinstruction.

The USING statement and load address (LA) instructions are used jointly to map the information table by means of the TC#CCOTP DSECT. This DSECT provides the labels (addresses) needed for further manipulation of the data that is placed in the information table, such as printing or displaying it. Note that the information table length may vary, depending on the number of device control tables required for the specific auxiliary devices attached.

```
1       10    16
        START Ø
        TU#DSTZ                          DUST Proc call
          .
          .
          .
        USING TC#CCOTP,9                 DUST DSECT COVER REGISTER
          .
          .
          .
CCACPY1 CCACPY TRMLST1, LTRMLST1, CCAINF1, LCCAINF1
        LA    9, CCAINF1                 Cover Table 1
          .
          .
          .   (BAL code to manipulate the information obtained)
CCACPY2 CCACPY TRMLST2, LTRMLST2, CCAINF2, LCCAINF2
        LA    9, CCAINF2                 Cover Table 2
          .
          .
          .   (BAL code to manipulate the information obtained)
CCACPY3 CCACPY TRMLST3, LTRMLST3, CCAINF3, LCCAINF3
        LA    9, CCAINF3                 Cover Table 3
          .
          .
          .   (BAL code to manipulate the information obtained)
        EOJ
WORK    DS    8F                         Storage Constants
TRMLST1 DC    C'TRM1'                    Terminal name list 1
        DC    X'FFFFFFFF'
```

(cont)

```
1         10    16
LTRMLST1 EQU   *-TRMLST1
TRMLST2  DC    C'TRM1'                  Terminal name list 2
         DC    C'TRM2'
         DC    X'FFFFFFFF'
LTRMLST2 EQU   *-TRMLST2
TRMLST3  DC    C'TRM1'                  Terminal name list 3
         DC    C'TRM2'
         DC    C'TRM3'
         DC    X'FFFFFFFF'
LTRMLST3 EQU   *-TRMLST3
CCAINF1  DS    6F                       Information Table 1
LCCAINF1 EQU   *-CCAINF
CCAINF2  DS    7F                       Information Table 2
LCCAINF2 EQU   *-CCAINF2
CCAINF3  DS    8F                       Information Table 3
LCCAINF3 EQU   *-CCAINF3
```

NOTE:

Work area for output requires extra four bytes for X'FFFFFFFF', which will be supplied by CCACPY macro as an end delimiter.

Example 3:

If the second form of the macroinstruction is used, the expansion of the macroinstruction is not performed. This form indicates that you have created your own CCACPY parameter list and have loaded the address of this parameter list into register 1. Note that the parameter list must be on a full-word boundary. This could be done as follows:

```
         .
         .
         .
LA       1, CPY1          Load CCACPY parameter list address
CCACPY (1)
         .
         .
         .
LA       1, CPY2          Load CCACPY parameter list address
CCACPY (1)
         .
         .
         .
LA       1, CPY3          Load CCACPY parameter list address
CCACPY (1)
         .
         .
         .
EOJ
```

(cont)

```
1         10      16
CPY1      DS      ØF          Full-word boundary
          DS      XL1         Storage for error code byte
          DS      XL1         Unused
          DS      XL1         Storage for function code byte
          DC      A(CCAINF1)  Full-word address of information table
          DC      A(TRMLST1)  Full-word address of terminal name list
          DC      H '24'      Length in bytes of CCAINF1
          DC      H '8'       Length in bytes of TRMLST1 (including
                              4 bytes for the delimiter in the terminal
                              name list (FFFFFFFF))
CPY2      DS      ØF
          DS      XL1
            .
            .
            .
          etc
            .
            .
            .
```

The terminal name lists and information tables are constructed the same as in example 1.

2.22.4.1.1. CCACPY Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the CCACPY macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the CCACPY parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-34 describes the error conditions detected during the execution of this macroinstruction.

Table 2-34. CCACPY Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QTNF	Terminal not found
	TQ#QTLG	Data beyond table or program area
	TQ#QTNFW	Parameter list not on full-word boundary
	TQ#QNDTH	Invalid work area length or no work area specified in parameter list

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

QCLEAR

2.22.4.2. Clear Designated Queues (QCLEAR)

Function:

Clears queues specified in the operand field of all messages without transmitting them. All associated network buffers are released. The function can apply to a single queue, the queues associated with a single terminal, LOCAP, or process file, or all queues associated with a line.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	QCLEAR	L,line-name P,{process-file-name / locap-name} [{LOW / MED / HIGH}] T,terminal-name [,{LOW / MED / HIGH}] (1)

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-21 and Table 2-35.

Parameters:

L

Clears all messages on all output queues associated with the line named in the line-name operand.

NOTE:

When multiple programs send messages to terminals connected to the same line in a global environment, one program can inadvertently clear messages placed on output queues by another program. In this case, it is desirable to clear queues individually by using the T operand.

P

Indicates that the clear function pertains to all the queues associated with a process file or to a specific priority queue of the file.

T

Indicates that the clear function pertains to all the queues associated with a specific terminal (terminal name) or to a specific priority queue of the specified terminal.

(1)

Indicates you have preloaded the address of your own parameter list in register 1. This parameter list must be on a full-word boundary and conform to the format in Figure 2-21 and Table 2-35.

line-name

Indicates the line associated with the queue or queues.

process-file-name

Identifies a process file associated with the queue or queues.

locap-name

Identifies the LOCAP associated with the queue or queues.

terminal-name

Identifies the terminal with the queue or queues.

{LOW / MED / HIGH}

Indicates the low, medium, or high priority queue of a specific process file, LOCAP, or terminal. If omitted, *all* queues associated with the specified name are cleared.

Example:

```
1         10    16
          QCLEAR L,LNE2
          QCLEAR (1)
          QCLEAR T,TRM3,HIGH
          QCLEAR P,PFILE1,LOW
```

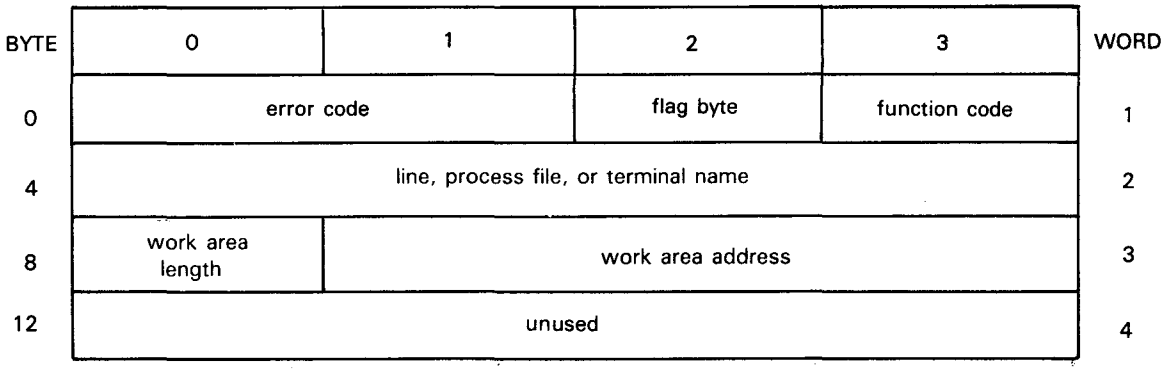


Figure 2-21. QDEPTH/QHOLD/QRELSE/QTRANS/QCLEAR Parameter List Functional Field Description

Table 2-35. QDEPTH/QHOLD/QRELSE/QTRANS/QCLEAR Parameter List Detailed Field Descriptions*

Byte	Field		Content	Comment	Word
	Label*	Type and Length			
0	TQ#QERR	H	Error flags	Set by ICAM	1
2	TQ#QFLAG	YL1	Type indicator	Differentiates P, L, and T types	
3	TQ#QFUNC	YL1	Function indicator	Indicates which macroinstruction was coded	
4	TQ#QPLTN	CL4	Process file, line, or terminal name		2
8	TQ#QWALG	YL1	Work area length in bytes	Used only by QDEPTH	3
9	TQ#QWORK	A3	Work area address	Used only by QDEPTH	

* For QTRANS functions, TQ#QFN1 and TQ#QFN2 are equivalents for TQ#QPLTN and TQ#QWALG corresponding to the two file names.

** DSECT TQ#QDSCT, which is part of TU#DSTZ.

2.22.4.2.1. QCLEAR Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the QCLEAR macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the QCLEAR parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-36 describes the error conditions detected during the execution of this macroinstruction.

Table 2-36. QCLEAR Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QLNF	Line not found
	TQ#QTNF	Terminal not found
	TQ#QTNFW	Parameter list is not on a full-word boundary.
	TQ#QNPF	Process file not found

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

QDEPTH

2.22.4.3. Queue Message Count (QDEPTH)

Function:

Indicates the number of messages on the queues of either a locap file, a process file, or a terminal. The number of messages in each queue is stored in a user-specified work area. The fields comprising this work area are illustrated in Figure 2-22 and Table 2-37.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	QDEPTH	P,{process-filename / locap-name},workarea-name, workarea-length T,terminal-name,workarea-name,workarea-length (1)

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-21 and Table 2-35.

Parameters:

P
Indicates that message counts are to be stored for the one to three queues associated with the specified process file (PRCS) or LOCAP.

T
Indicates that message counts are to be stored for all the output queues associated with specified terminal.

(1)
Indicates you have preloaded the address of your own parameter list in register 1 that identifies the work area and the queue to be examined. This parameter list must be on a full-word boundary and conform to the format in Figure 2-21 and Table 2-35.

process-filename
Identifies the process file for which the message counts are to be obtained.

`locap-name`
Identifies the LOCAP for which the message counts are to be obtained.

`terminal-name`
Identifies the terminal where the queue message counts are to be obtained.

`workarea-name`
Identifies the label of the area in your program where the message counts are to be stored. This work area must be on a half-word boundary.

`workarea-length`
Indicates the work area length in bytes.

Example:

```
1         10    16
-------------------------------------
          QDEPTH P,PRF3,WORK,WALEN
          QDEPTH (1)
```

BYTE	0	1	2	3	WORD
0	work area length		number of queues		1
4	name specified in QDEPTH macroinstruction				2
8	internal name of low priority queue				3
12	number of messages on low priority queue		internal name of medium priority queue (1st half)		4
16	internal name of medium priority queue (2nd half)		number of messages on medium priority queue		5
20	internal name of high priority queue				6
24	number of messages on high priority queue		name of intercept queue* (1st half)		7
28	name of intercept queue* (2nd half)		number of messages on* intercept queue		8

* Intercept queues are created only for terminals; not used for process files or locap files.

Figure 2-22. QDEPTH Work Area Field Descriptions

Table 2-37. QDEPTH Work Area Detailed Field Descriptions

Byte	Field		Usage	
	Label	Type and Length	Process File or LOCAP (P)	Terminal (T)
0	TQ#QWAL	H	Number of bytes in work area	Same
2	TQ#QNQ	H	Number of queues - 1, 2, or 3	3
4	TQ#QNAM	CL4	PRCS or LOCAP name specified	TERM name specified
8	TQ#QLOW	CL4	Internal name of low priority queue	
12	TQ#QLCNT	H	Number of messages on low priority queue	
14	TQ#QMED	CL4	Internal name of medium priority queue	
18	TQ#QMCNT	H	Number of messages on medium priority queue, if present	
20	TQ#QHIGH	CL4	Internal name of high priority queue	
24	TQ#QHCNT	H	Number of messages on high priority queue	
26	TQ#QINTR	CL4	(Not used or not required)	Name of intercept queue, if any; defaults to 00_{16} if an intercept queue is not defined.
30	TQ#QICNT	H		Number of messages on intercept queue, or hexadecimal zeros.

NOTES:

1. The name of the DSECT for this table is TQ#QDWA (expand macroinstruction TU#DSTZ).
2. Work area must be aligned on a half-word boundary.

If you specify less than three queues on the TERM, PRCS, or LOCAP macroinstructions (HIGH, MEDIUM, and LOW operands), any queue not specified is automatically defaulted. That is, ICAM provides linkages to existing queues even though you did not request them.

When you issue a QDEPTH macroinstruction, ICAM returns a message count in all of the message count areas whether that queue was provided by default or not. The count that appears in each message count work area depends on how each queue was created (by default or by request) and how the other queues were created. Table 2-38 shows how queues are defaulted and the counts you might expect as a result of a QDEPTH request.

The left side of Table 2-38 shows the combinations of queues you might specify in a network definition; the right side of the table shows how each queue is provided - by default or by definition. YES indicates that the queue was provided by default; NO indicates you specified the queue. L, M, and H characters indicate the source of the

count that would be present. For example, if you only specified a low priority queue in your network definition, and then issued a QDEPTH request, the count in the medium priority and high priority work areas would be the same as in the low priority work area. If you specified a low priority queue and high priority queue at network definition time (but not a medium priority queue), then the message count returned in the medium priority queue would be equal to that found for the high priority queue.

Table 2-38. Relationship of Queues Generated and Message Count Returned for QDEPTH

Queue Specified in Network			Queue Generated and Default Priority Level					
LOW	MED	HIGH	LOW	Was Queue Defaulted?	MED	Was Queue Defaulted?	HIGH	Was Queue Defaulted?
NO	NO	NO	L	YES	L	YES	L	YES
NO	NO	YES	H	YES	H	YES	H	NO
NO	YES	NO	M	YES	M	NO	M	YES
NO	YES	YES	M	YES	M	NO	H	NO
YES	NO	NO	L	NO	L	YES	L	YES
YES	NO	YES	L	NO	H	YES	H	NO
YES	YES	NO	L	NO	M	NO	M	YES
YES	YES	YES	L	NO	M	NO	H	NO

NOTE:

Count returned is equal to that found on:

L = Low priority queue
M = Medium priority queue
H = High priority queue

2.22.4.3.1. QDEPTH Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the QDEPTH macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the QDEPTH parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-39 describes the error conditions detected during the execution of this macroinstruction.

Table 2-39. QDEPTH Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QTNF	Terminal not found
	TQ#QNPF	Process file not found
	TQ#QTLG	Data beyond table or program area
	TQ#QTNFW	Parameter list not on full-word boundary
	TQ#QNDTH	Invalid work area length or no work area in parameter list

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

QHOLD

2.22.4.4. Hold Output Transmission (QHOLD)

Function:

Suspends transmission of output messages from a queue or group of queues. The suspension can apply to a single queue, the queues associated with a single terminal, or all the queues associated with a line. The suspension remains in effect until a QRELSE macroinstruction is issued by your program or a UP command is issued by the console operator. The user program (or the MPPS) may continue to transfer messages to a suspended queue.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	QHOLD	{ T,terminal-name [,{HIGH / LOW / MED}] / L,line-name / (1) }

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-21 and Table 2-35.

Parameters:

T

Indicates that the hold function pertains to all the queues associated with a specific terminal (terminal name), if no priority level is specified, or to a specific priority queue of the specified terminal.

L

Indicates that the hold function pertains to all the queues associated with all the terminals defined as part of the specified line (line name).

(1)

Indicates you have preloaded the address of your own parameter list in register 1. This parameter list must be on a full-word boundary and conform to the format in Figure 2-21 and Table 2-35.

`line-name`
Identifies the line associated with the queue or queues.

`terminal-name`
Identifies the terminal with the queue or queues.

`{HIGH / LOW / MED}`
Indicates that the specified terminal priority queue is to be suspended.

Example:

```
1         10    16
-------------------------------------------
          QHOLD L,LNE2
          QHOLD (1)
          QHOLD T,TRM3,HIGH
```

2.22.4.4.1. QHOLD Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstruction and an error occurs during macroinstruction processing, control returns inline to the address following the QHOLD macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the QHOLD parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-40 describes the error conditions detected during the execution of this macroinstruction.

Table 2-40. QHOLD Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QLNF	Line not found
	TQ#QTNF	Terminal not found
	TQ#QTNFW	Parameter list is not on a full-word boundary.

NOTE: The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

QRELSE

2.22.4.5. Resume Output Transmission (QRELSE)

Function:

Allows transmission of output messages from a previously held queue or group of queues.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	QRELSE	{T,terminal-name [,{HIGH / LOW / MED}]} {L,line-name / (1)}

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-21 and Table 2-35.

Parameters:

T
Indicates that, if no priority level is specified, all the queues associated with a specific terminal are to be released, or if a priority is specified, just those queues with that specific priority of the named terminal are to be released.

L
Indicates that all the queues associated with a specific line are to be released.

(1)
Indicates you have preloaded the address of your own parameter list in register 1 that identifies the queue or queues to be released. This parameter list must be on a full-word boundary and conform to the format in Figure 2-21 and Table 2-35.

line-name
Identifies the line associated with the queue or queues.

`terminal-name`
Identifies the terminal associated with the queue or queues.

`{HIGH / LOW / MED}`
Indicates that the specified priority queue is to be released.

Example:

```
1         10    16
          QRELSE L,LNE3
          QRELSE T,TRM6
          QRELSE (1)
          QRELSE T,TRM3,HIGH
```

2.22.4.5.1. QRELSE Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the QRELSE macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the QRELSE parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-41 describes the error conditions detected during the execution of this macroinstruction.

Table 2-41. QRELSE Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QLNF	Line not found
	TQ#QTNF	Terminal not found

NOTE:
The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

2.22.4.6. Transfer Message Queues (QTRANS)

Function:

Transfers the messages from a queue of one process file, LOCAP, or terminal to another. If positional parameters 3 and 4 are omitted, each message of positional parameter 1 is queued to the destination specified in the message header. If positional parameter 3 is given, each message of positional parameter 1 will be placed on the corresponding priority queue of positional parameter 3. The parameter list table is described in Figure 2-21 and Table 2-35. QTRANS can only be used for main storage queues, not disk queues.

A special option of QTRANS may be used to output messages previously stored on a process file for later transmission.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	QTRANS	{{filename$_1$ [,{ALL / HIGH / LOW / MED}]} / (1)} filename$_2$ [,{HIGH / LOW / MED}]

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-21 and Table 2-35.

Parameters:

filename$_1$
: Specifies the name of the process file, LOCAP, or terminal queue from which messages are transferred.

(1)
: Indicates you have preloaded the address of your own parameter list in register 1. This parameter list must be on a full-word boundary and conform to the format in Figure 2-21 and Table 2-35.

{ALL / HIGH / LOW / MED}
: Indicates from which priority queue messages are to be transferred. Default is ALL.

`filename2`

Specifies the name of the process file, terminal queue, DLIST, or LOCAP to which messages are transferred.

`{HIGH / LOW / MED}`

Indicates priority queue messages. Default causes messages to be queued to the corresponding priority queue from which it was transferred.

NOTE:

A special option of QTRANS may be used to remove messages previously stored on a process file and output them to their final destination. When this option is used, only the first two parameters are used as shown in Example 2. The process file is used as a floating queue to hold messages until they are to be transmitted. The following steps are required:

1. *Set PUTCP processing flag TM#DFQR by executing:*

   OI TM#DDTF,TM#DFQR

2. *Move the name of the desired final destination of the message into the destination queue-name field of the PUTCP packet (TM#DNAM).*

3. *Execute a PUTCP to a process file with the name of the process file being used as the floating queue specified in the TM#DENA destination routing name field.*

4. *Repeat steps 2 and 3 for each message to be placed on the floating queue.*

5. *When the messages are being transmitted, execute:*

   QTRANS process-file,priority-level

Examples:

```
  1         10    16
1.|         QTRANS TRM2,MED,TRM5,LOW
2.|         QTRANS PRC2,ALL
```

1. The user transfers a message from the medium priority queue of the process file named TRM2 to the low priority queue of the process file named TRM5.

2. The user causes all messages referenced on the floating queue named PRC2 to be transmitted.

2.22.4.6.1. QTRANS Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the QTRANS macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE.

Register 1 contains the address of the QTRANS parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-42 describes the error conditions detected during the execution of this macroinstruction.

Table 2-42. QTRANS Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QLNF	Line not found
	TQ#QTNF	Terminal not found
	TQ#QNPF	Process file not found
	TQ#QIQT	Invalid queue priority

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

RELEASM

2.22.4.7. Reset Intercept Queue Bit (RELEASM)

Function:

Resets the intercept queue hold bit in an intercept queue control table. This releases intercepted messages and permits transmissions to that terminal to resume. This instruction is used in conjunction with the INTERCEPT MPPS macroinstruction.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	RELEASM	{terminal-name} {(1)}

This macroinstruction may be written in two formats. In the first format, the macroinstruction expansion creates the parameter list inline for you and inserts the operands you specify. In the second format, using register notation, you optionally elect to build your own parameter list somewhere in your nonexecutable code and have preloaded the address of this list into the register. In this second case, you must ensure that the parameter list is on a full-word boundary and the format is as shown in Figure 2-23 and Table 2-43.

Parameters:

terminal-name
Identifies the terminal from which the intercept queue is to be released.

(1)
Indicates you have preloaded the address of your own parameter list in register 1. This parameter list must be on a full-word boundary and conform to the format in Figure 2-23 and Table 2-43.

Example:

```
1          10     16
           RELEASM TRM5
```

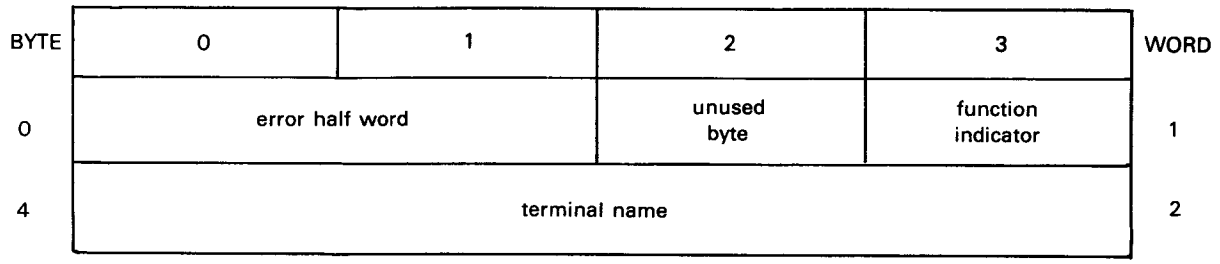


Figure 2-23. RELEASM Parameter List Functional Field Description

Table 2-43. RELEASM Parameter List Detailed Field Descriptions

Byte	Field		Content	Word
	Label	Type and Length		
0	TQ#QERR	H	Error half word	1
2		XL1	Not used	
3	TQ#QFUNC	XL1	Function indicator	
4	TQ#QPLTN	CL4	Terminal name	2

2.22.4.7.1. RELEASM Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the RELEASM macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE.

Register 1 contains the address of the RELEASM parameter table. Register 2 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-44 describes the error conditions detected during the execution of this macroinstruction.

Table 2-44. RELEASM Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QTNF	Terminal not found

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

TRMREP

2.22.4.8. Change Telephone Number (TRMREP)

Function:

Changes the phone number in the line control table.

Format:

LABEL	△OPERATION△	OPERAND
[symbol]	TRMREP	line-name,terminal-name,workarea-address ,FIELDS=(CALL)

Parameters:

`line-name`
Identifies the line to which the subject terminal is connected.

`terminal-name`
Identifies the subject terminal.

`workarea-address`
Address of the work area in your program that contains the replacement phone number. The old number is from the CALL parameter of the network definition LINE macroinstruction.

The phone number information in the work area starts at byte 29. (See Figure 2-24.) The first byte of this information contains the total count, in binary, of the number of dialing digits in the phone number. The phone number then follows in decimal characters.

The number of characters in the new number must be equal to or less than the number of characters in the number generated originally at system generation time and must not exceed 48.

When autodialing is used, include a hyphen in the phone number to cause a 1.1-second pause. This delay allows time for a connection to be made before more dial characters are sent.

`FIELDS=(CALL)`
Indicates the replacement of the phone number in the line control table phone directory with a new number from the TRMREP work area.

Example:

```
1         10    16
          TRMREP  LØ14,TØØ3,TRWA,FIELDS=(CALL)
```

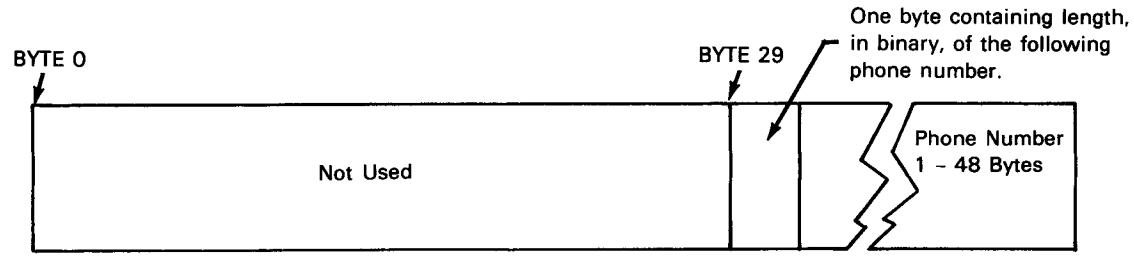


Figure 2—24. TRMREP Work Area Functional Field Description

NOTE:

Always issue the TRMREP macroinstruction to an inactive line after network activation, that is, at any time before issuing a LNEREQ macroinstruction to an inactive line or after issuing a LNEREL macroinstruction to an active line.

2.22.4.8.1. TRMREP Error Processing

When you specify DUSTERR=INLINE in the CCA and LOCAP macroinstructions and an error occurs during macroinstruction processing, control returns inline to the address following the TRMREP macroinstruction. When you don't specify DUSTERR=INLINE, errors are returned at the specified NETREQ/NATTACH ERRET= address. However, we recommend that you code your program for inline error returns and specify DUSTERR=INLINE. (See 2.12 for details.)

Register 1 contains the address of the TRMREP parameter table. Register 0 contains:

- zeros in bytes 0, 1, and 3; and
- TQ#QERR error codes in byte 2.

Table 2-45 describes the error conditions detected during the execution of this macroinstruction.

Table 2-45. TRMREP Error Conditions

Byte	Error Code	Cause/Condition
TQ#QERR	TQ#QLNF	Line not found
	TQ#QTNF	Terminal not found
	TQ#QTLG	Data beyond table or program area
	TQ#QNDTH	Phone number too long

NOTE:

The DSECT for this table is TQ#QDSCT, which is part of TU#DSTZ.

2.23. SAMPLE PROGRAMS THAT USE ICAM

It is a difficult task for any programmer to work with an unfamiliar product because of new terms and required programming techniques. This section attempts to alleviate this problem by providing some uncomplicated examples that are fully explained and fully tested.

Each program in this section was executed as described in the accompanying text. If you have not run ICAM before, we welcome you to run these examples on your computer. Don't forget to change the things that are different for your site, such as the addresses for your terminals and the telephone number of the communications adapter or single line communications adapter port. Also, make certain you have the facilities you need, both hardware and software, included in your system. For example, you must include an automatic dialer if you are going to do automatic dialing.

The examples provided are purposely elementary so as not to cloud the programs with extensive error analysis and data processing. Another reason the examples are elementary is that ICAM's portion of any applications program is usually small because ICAM is devoted to handling communications only. Only you can decide exactly what your program needs to do once ICAM has delivered a message to it or sent one to a terminal. The task of processing data is left to other parts of your program, perhaps in conjunction with the Sperry Univac information management system (IMS).

2.23.1. How to Write a Program to Use ICAM in a Dedicated Network Environment

In an ICAM dedicated network environment, all of the communications lines and terminals described in a network definition are dedicated to the program that successfully requests that network. Each network typically has terminals associated with a particular program, and programs cannot share lines and terminals concurrently.

The following two examples show some basics on how to use PUTCP, GETCP, and DTFCP macroinstructions in your program to communicate with terminals and other communications end users. To do this, we need to show the most basic dedicated network definition possible: one communications line with one terminal attached to it. Such a network definition is shown in Figure 2-25 along with a picture that illustrates the configuration. We named our network NET1.

Note that we are not trying to explain how to define networks here; the only reason we show this and any of the other network definitions in this book is to show the relationship of a network definition and the statements in a user program. For complete information on how to create network definitions, refer to the current version of the ICAM network definitions and operations user guide, UP-8947.

```
NET1     CCA      TYPE=(STDMCP)
         BUFFERS  35,64,3,ARP=24
LNE1     LINE     DEVICE=(UNISCOPE),CALL=(1234567),                    X
                  TYPE=(2000,SWCH,SYNC),INPUT=(PRF1)
TRM1     TERM     FEATURES=(U400),ADDR=(24,55),LOW=MAIN
PRF1     PRCS     LOW=MAIN
         ENDCCA
```

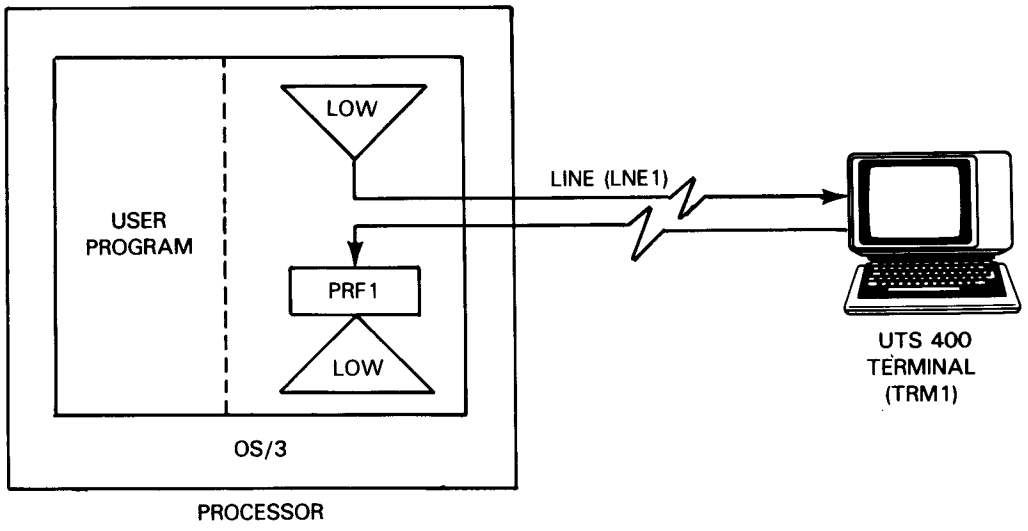


Figure 2-25. Basic ICAM Network Definition

In Figure 2-25, notice that network NET1 defines a terminal named TRM1 connected to a communications line named LNE1, and a process file named PRF1. The ICAM network includes a low priority output queue related to TRM1 and a low priority input queue that is related to process file PRF1. In the following examples, our program sends output messages to terminal TRM1 and retrieves input messages from process file PRF1.

When we issue a PUTCP to send a message to terminal TRM1, ICAM copies the message from our program and places it on the output queue related to TRM1. ICAM sends the message as soon as it can. As input messages arrive from terminal TRM1, ICAM queues them to process file PRF1. Our program issued GETCP requests to PRF1 to obtain these messages.

Referring to Figure 2-26, we show how to organize a program to use ICAM. Please notice that this figure is only meant to show a general organization of a program you might write; there are additional macroinstructions you need to include. Later on, we will present a complete working program that uses all of the basics and is fully annotated. Following Figure 2-26 are a number of comments that explain why each statement was used.

```
1         10    16
           START
             .
             .  Your applications code
             .
          BAL        12,UWRITE1
             .
             .  Your applications code
             .
UWRITE1   NETREQ     NET1,LNEREQ=(YES)
             .
             .  Your applications code
             .
APUT      PUTCP      Q00001,WORKOUT
             .
             .  Your applications code
             .
GET1      GETCP      PRF1,WORKIN
             .
          NOPUT      Your applications code
             .
             .
             .
NOGET        .  Your applications code
             .
             .
NMSG         .  Your applications code
             .
RELEASE   NETREL     NET1
             .
             .  Your applications code
             .
          EOJ
          ØF
Q00001    DTFCP      TYPE=PT,DEST=(T,TRM1),ERRET=NOPUT
PRF1      DTFCP      TYPE=GT,LEVEL=LOW,NOMAV=NMSG,ERRET=NOGET
WORKOUT   DC CL256'△'
WORKIN    DC CL256'△'
END
```

Figure 2-26. How to Organize a Basic User Program

- How to request and release a network

  Before we can use ICAM, we must request the ICAM network we are going to use. ICAM is loaded into OS/3 as a symbiont, and our program must issue a network request (NETREQ) macroinstruction to obtain a dedicated network from the symbiont. Our program must request a network before any communications can begin. When our program is finished with ICAM, it releases the network with a network release (NETREL) macroinstruction.

  For example, to request network NET1 we issue:

```
label NETREQ NET1,LNEREQ=(YES)
```

  This statement obtains the network named NET1 from the symbiont and tells ICAM to activate all communications lines in that network.

  To release network NET1 when our program is all finished with it, we issue:

```
label NETREL NET1
```

- How to control the data

  We must define control file information relating to the various PUTCP and GETCP requests we will use. We do this with DTFCP macroinstructions. (Note that DTFCP macroinstructions are declarative and they must be placed with the other declarative macroinstructions in your program.) DTFCP macroinstructions tell ICAM how to handle a message.

  When we want to send a message to terminal TRM1, we define a DTFCP with TYPE=PT that looks like this:

```
Q001 DTFCP TYPE=PT,DEST=(T,TRM1),ERRET=NOPUT
```

  This means: we named the DTFCP Q001; specified that it was related to an output (PT) function; that the data is to go to a terminal named TRM1; and, in case of an error, control must go to an error routine at address NOPUT in our program.

  When we want to get messages from PRF1, we define a DTFCP with TYPE=GT as follows:

```
PRF1 DTFCP TYPE=GT,LEVEL=LOW,NOMAV=NMSG,ERRET=NOGET
```

  This means: we want to get a message from a queue on process file PRF1 (the label defines the input source); this DTFCP is related to one input function (GT); ICAM is to access the low priority queue on the process file; control is to return to a routine in our program named NMSG if no message is available; and, in case of an error, we want control to return to an error processing routine in our program named NOGET.

- How to send or receive a message

  Now that we have defined the DTFCP input and output control files, we must code the actual PUTCP and GETCP macroinstructions to initiate the movement of messages (data) between our program and ICAM. Each GETCP or PUTCP points to the address of a related DTFCP and to a work area in our program. For output, this work area is where the data to be sent is located; for input, this is where you want ICAM to place input data it is holding on a queue for you. To send a message to some destination, we use a PUTCP macroinstruction such as:

```
PUT1 PUTCP Q001,WORKOUT
```

  This PUTCP sends data located in your program at (address) WORKOUT to a destination defined and controlled by a DTFCP named Q001.

  To get a message held by ICAM on a queue for us, we use a GETCP macroinstruction such as:

```
GET1 GETCP PRF1,WORKIN
```

  This GETCP instructs ICAM to move a message queued on process file PRF1 to a work area located in your program at an address named WORKIN.

- What to do about errors

  One other thing we must do in any program is to have some kind of error processing, just in case something goes wrong. In our example, we just issued a message to the operator. When you write your own program, you probably would need to do a number of other things, such as: determining the cause of the error by interrogating the error fields in the related DTFCP that ICAM sets in such cases; dumping your program; releasing lines; and any number of other things unique to your own situation.

- How to communicate with an auxiliary device

  In the previous example, the message destination was to the primary device at terminal TRM1. This is usually the display screen of the terminal system. However, if we want to send a message to an auxiliary device, such as a communications output printer (COP) or a tape cassette system, we must define the auxiliary device in our ICAM network definition. Figure 2-27 shows a network we modified to define auxiliary devices, named NET2. The only difference between networks NET1 and NET2 is the addition of the AUXn suboperands.

```
NET2     CCA       TYPE=(STDMCP)
         BUFFERS   35,64,3,ARP=24
LNE1     LINE      DEVICE=(UNISCOPE),CALL=1234567,                X
                   TYPE=(2000,SWCH,SYNC),INPUT=(PRF1)
TRM1     TERM      FEATURES=(U400),ADDR=(24,55),                  X
                   AUX1=(COP,73),                                 X
                   AUX2=(TCS,74,75),                              X
                   AUX3=(TCS,76,77),                              X
                   LOW=MAIN
PRF1     PRCS      LOW=MAIN
         ENDCCA
```

Figure 2-27. A Basic Network Definition that Defines Auxiliary Devices

Once we have prepared our network to support auxiliary devices, we modify the DTFCP in our program to send a message to the auxiliary device.

When we want our program to send a message to an auxiliary device, we must set up a DTFCP in our program to address the auxiliary device before we issue a PUTCP. There are three steps involved:

1. Move the auxiliary device number into field TM#DDVC of the output DTFCP. (See the AUXn operand of the TERM macroinstruction in the current version of the ICAM network definition and operations user guide, UP-8947 for details on how to specify auxiliary device numbers.) For example, to send data to the first tape cassette/disk drive on TRM1, specify:

   ```
   label MVI TM#DDVC,2
   ```

2. Move the auxiliary device function code into field TM#DSPEC of the DTFCP. For example, to write one block of data, specify:

   ```
   label MVI TM#DSPEC,TM#DADWR
   ```

   We specify a name (TM#DADWR) instead of using the actual hexadecimal value for the function; if a later release of ICAM changes the value for this function, all we need to do is reassemble our program and the assembler will insert the correct value for us. All of the labels you can use are supplied and explained along with the DTFCP macroinstruction explanations in this user guide.

3. Specify in field TM#DSEG that the message is going to an auxiliary device and whether this is the start of a new message or the end of the current message. For example, if we are sending a complete message (the first block is also the last), specify:

   ```
   label OI TM#DSEG,TM#DAFN++TM#DHDR++TM#DTND
   ```

   Notice that the ++ symbols between each of the function names causes them to be ORed together, thus setting several bits in TM#DSEG.

Figure 2-28 is an example of how to send a message to an auxiliary device. It uses the instruction just explained. We shaded the areas that involve the auxiliary devices.

Also, notice the use of a dummy section (DSECT) in this example to map a DTFCP. At the beginning of the program, we defined a macroinstruction named TM#DEST with operand DEST; base register 2 is assigned to the dsect. When we assemble our program, this macroinstruction assembles all the labels required to map an output DTFCP into our program. When we need to move values into our DTFCP area named Q001, all we need to do is load the address of the output DTFCP into register 2; we can use simple move instructions with tag names to set up our DTFCP.

```
          START
          TM#DSECT DEST                          USER DSECT
          USING TM#DEST,2                        SET COVER REGISTER
            }
            }    Applications code
            }
          LA    2,Q00001                         MAP DESTINATION TABLE
          MVI   TM#DDVC,2                        SET AUXILIARY DEVICE
          MVI   TM#DSPEC,TM#DADWR                SET AUX DEVICE COMMAND
          OI    TM#DSEG,TM#DAFN++TM#DHDR++TM#DTND SET FOR COMPLETE MESSAGE
                                                 (AND INCLUDE AUX DEVICE INFO)
          BAL   12,UWRITE1
            }
            }    Applications BAL code
            }
UWRITE1   NETREQ NET1,LNEREQ=(YES)
                  .
                  .
                  .
          PUTCP Q00001,WORKOUT
            }
            }    Applications BAL code
            }
             .
             .
             .
          NOPUT OPR MSG,12
             .
             .
             .
          EOJ
          DS    0F
Q00001    DTFCP TYPE=PT,DEST=(T,TRM1),ERRET=NOPUT
          DS    0H
          DC    H'256'
          DC    CL256'△'
          END
```

Figure 2-28. Basic User Program Showing How to Communicate with Auxiliary Devices

We have just shown how to use ICAM in a very elementary way, that is, how to use PUTCP, GETCP, and DTFCP and how to call a few dsects. In the following text, we present some working programs. These programs are also elementary, but they do execute. They should be helpful to you if you have not run ICAM before.

2.23.2. A Working User Program

Figure 2-29 graphically illustrates a processor running ICAM. This ICAM network controls a single communications line with only one associated terminal. The terminal is a SPERRY UNIVAC UTS 400 Universal Terminal System (UTS 400) terminal; no auxiliary devices are connected to the UTS 400 for this example. The network is named NPR and the user program is named USERV9. Figure 2-30 shows how to define and code a network definition for network NPR. Refer to the current version of the ICAM network definition and operations user guide, UP-8947 for details on how to create ICAM symbiont networks.

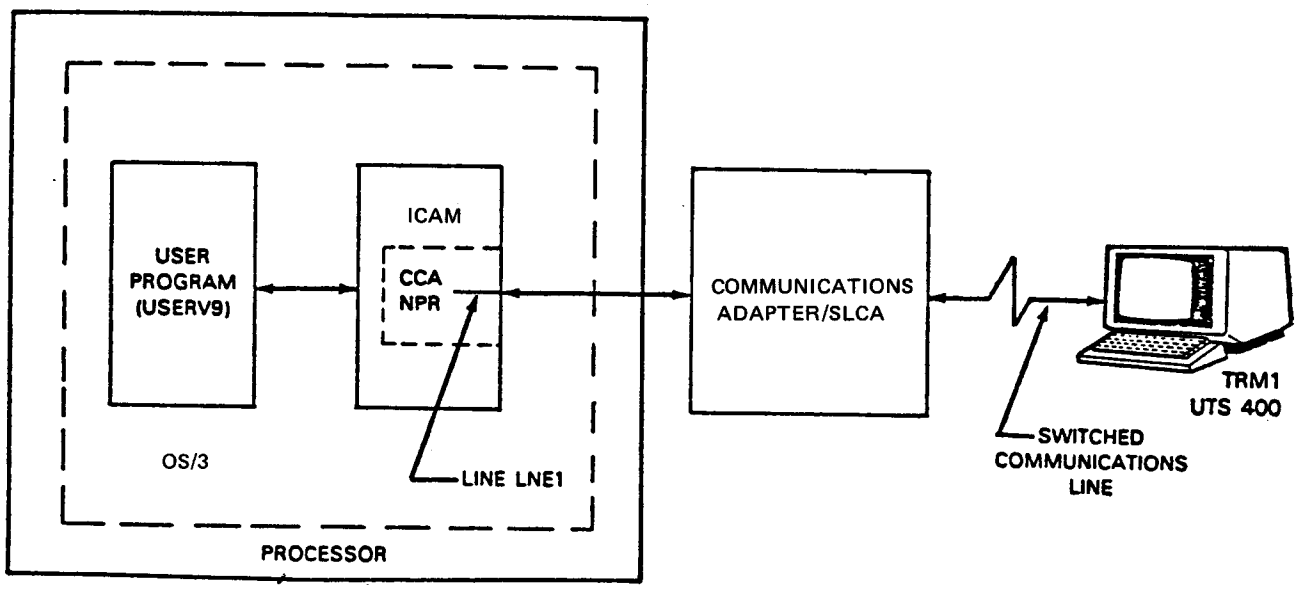


Figure 2-29. Elementary Communications System Using ICAM

```
1         10    16                                                  72
NPR       CCA     TYPE=(STDMCP)
          BUFFERS 20,100,,ARP=12
LNE1      LINE    DEVICE=(UNISCOPE),                                  C
                  TYPE=(2000,SWCH,SYNC,UNAT),                         C
                  ID=8,                                               C
                  INPUT=PRFA,                                         C
                  LOW=MAIN
TRM1      TERM    ADDR=(28,51),                                       C
                  FEATURES=(U400,1920)
PRFA      PRCS    LOW=MAIN
          ENDCCA
```

Figure 2-30. Coding for an Elementary Network Named NPR

2.23.2.1. An Elementary Network Named NPR

Let's look at Figure 2-30 and then discuss why we coded each statement the way we did.

CCA Macroinstruction:

The CCA macroinstruction indicates the beginning of a network definition. We named this network NPR, but any 1- to 4-character label would be all right as long as it is unique. Once you name your network, however, you must use its exact name whenever you refer to it in your program, for example, when your program issues a NETREQ macroinstruction to request the loading of the network.

TYPE=(STDMCP) means that this network provides the standard interface to your program.

BUFFERS Macroinstruction:

The BUFFERS macroinstruction specifies the system resources that are required; here it specifies 20 network buffers. Each network buffer is 100 words long (400 bytes). Twelve activity request packets (ARPs) are requested. ICAM uses activity request packets to pass parameters between the different ICAM modules.

LINE Macroinstruction:

The LINE macroinstruction defines our only communications line. If this was a larger system, we would probably have many lines and many LINE macroinstructions defined. We labeled this line LNE1, but you can call it any 1 to 4-character name you want as long as it is unique.

DEVICE=(UNISCOPE) means that this line uses the ICAM UNISCOPE remote device handler.

TYPE= specifies that the line speed is 2000 bits per second and that it is a dialed (SWCH) line. The line is synchronous (SYNC) and unattended (UNAT). Unattended means that when your program requests this network, ICAM conditions itself to automatically answer incoming calls to the computer. You can find the telephone numbers and the related communications adapter/single line communications adapter port for your system by asking your OS/3 system administrator.

INPUT=PRFA means that all incoming messages from any terminal on this line are directed to process file PRFA.

LOW=MAIN creates a single output queue in main storage for ICAM. ICAM uses the output queue for temporary storage of output messages until it can make the necessary connections (for example, dialing the telephone number of the remote terminal) and send any messages that are queued.

TERM Macroinstruction:

You use the TERM macroinstruction to define the characteristics of any terminals on a line. We labeled this terminal TRM1, but you can call it any unique 1- to 4-character name you want.

ADDR=(28,51) specifies the terminal's address (rid 28, sid 51) as wired into the terminal.

FEATURES=(U400,1920) tells ICAM that this terminal is a UTS 400 terminal with a 1920-character display screen.

PRCS Macroinstruction:

There are many uses for process files. In this case, we are using a process file to temporarily store input messages received by ICAM until our program can access them. Notice that this label (PRFA) agrees with the label we specified in the INPUT operand of the LINE macroinstruction.

LOW=MAIN specifies a low priority queue on this process file.

ENDCCA Macroinstruction:

ENDCCA indicates the end of the network definition.

2.23.2.2. A User Program that Uses Network NPR

The program shown in Figure 2-31 was executed successfully with the network definition shown in Figure 2-30. Our program is an elementary one designed to introduce you to ICAM.

Our program uses the network named NPR to send the following two messages:

```
WELCOME TO OS/3 ICAM
TYPE IN YOUR NAME PLEASE
```

It then issues a deferred GETCP request for input and yields control. When a message is received, the program places *THANK YOU* in front of it, sends it back to the terminal, and repeats the welcome messages. This process continues until the first word of a message is either the capital letters *DONE* or the lowercase letters *done*.

```
@P
   1.0000 USROS3   START 0
   2.0000          TN#DSECT DUST
   3.0000          TM#DSECT GETPUT
   4.0000          BALR  10,0
   5.0000          USING *,10
   6.0000 *                ********************************
   7.0000 *                ****** COVER INPUT DTFCP ******
   8.0000 *                ********************************
   9.0000          USING TM#PRCS,2
  10.0000          LA    2,PRFA
  11.0000 *                ********************************
  12.0000 *                ****** COVER OUTPUT DTFCP *****
  13.0000 *                ********************************
  14.0000          USING TM#DEST,3
  15.0000          LA    3,OTPT
  16.0000 *                ********************************
  17.0000 *                ****** REQUEST A NETWORK *******
  18.0000 *                ********************************
  19.0000 BEGIN    NETREQ NPR,ERRET=NETERR
+
  20.0000 *                ********************************
  21.0000 *                ********* SET UP PUTCP ********
  22.0000 *                ********************************
  23.0000 LOOP     OI    TM#DSEG,TM#DHDR++TM#DTND
  24.0000          XC    TM#DERR,TM#DERR
  25.0000 PUTIT1   PUTCP OTPT,MSGOT1
  26.0000 *                ********************************
  27.0000 *                ********* SET UP PUTCP ********
  28.0000 *                ********************************
  29.0000          OI    TM#DSEG,TM#DHDR++TM#DTND
  30.0000          XC    TM#DERR,TM#DERR
  31.0000 PUTIT2   PUTCP OTPT,MSGOT2
  32.0000 *                ********************************
  33.0000 *                **** SET UP DEFERRED GETCP ****
  34.0000 *                ********************************
  35.0000          XC    TM#PERR,TM#PERR
  36.0000          MVI   TM#PIND,TM#PIRL
  37.0000          MVC   TM#PCMPA,=A(BINGO)
  38.0000          MVC   MSGIN,TXTCHARS

  39.0000 GETIT    GETCP PRFA,MSGIN
  40.0000 *                ********************************
  41.0000 *                ** YIELD WITH NOTHING TO DO ***
  42.0000 *                ********************************
  43.0000          CYIELD
  44.0000 *                ********************************
  45.0000 *                **** WHAT KIND OF MESSAGE? ****
  46.0000 *                ****    DONE OR A NAME?    ****
  47.0000 *                ********************************
  48.0000 BINGO    CLC   TXTIN(4),=C'DONE'
  49.0000          BE    ENDJOB
  50.0000          CLC   TXTIN(4),=X'84969585'
  51.0000          BE    ENDJOB
  52.0000 *                ********************************
  53.0000 *                ********* SET UP PUTCP ********
  54.0000 *                ********************************
  55.0000          MVC   TXTOUT,TXTIN
  56.0000          MVC   TXTIN,C' '
```

Figure 2-31. Sample User Program that Uses Network NPR (Part 1 of 2)

```
  57.0000           MVC    TXTIN+1(L'TXTIN-1),TXTIN
+
  58.0000           OI     TM#DSEG,TM#DHDR++TM#OTND
  59.0000 PUTIT3    PUTCP  OTPT,MSGOUT
  60.0000           B      LOOP
  61.0000 ENDJOB    NETREL NPR
  62.0000           B      DONE
  63.0000 NETERR    LNEREL LNE1
  64.0000           LNEREQ LNE1
  65.0000 *                 **********************************
  66.0000 *                 *****  YIELD TO ICAM  **********
  67.0000 *                 *****  DEFERRED GET STILL  ****
  68.0000 *                 *****  PENDING          **********
  69.0000 *                 **********************************
  70.0000           CYIELD
  71.0000 NOGET     LA     1,MSG2
  72.0000           OPR    MSG2,12
  73.0000           B      DONEJOB
  74.0000 NOPUT     LA     1,MSG3
  75.0000           OPR    MSG3,12
  76.0000           B      DONEJOB

  77.0000 NOBUFF    LA     1,MSG4
  78.0000           OPR    MSG4,19
  79.0000 DONEJOB   SNAP   BEGIN,TXTCHARS
  80.0000 DONE      EOJ
  81.0000 *                 **********************************
  82.0000 *                 ****      CONSTANT AREA      ****
  83.0000 *                 **********************************
  84.0000 PRFA      DTFCP  TYPE=GT,ERRET=NOGET
  85.0000 OTPT      DTFCP  TYPE=PT,ERRET=NOPUT,DEST=(T,TRM1),NOBAV=NOBUFF
  86.0000           DS     0F
  87.0000 MSG1      DC     CL26'ERROR ISSUING NETREQ MACRO'
  88.0000 MSG2      DC     CL12'ERROR ON GET'
  89.0000 MSG3      DC     CL12'ERROR ON PUT'
  90.0000 MSG4      DC     CL19'NO BUFFER AVAILABLE'
  91.0000           DS     0H
  92.0000 MSGOT1    DC     X'0015'
  93.0000           DC     C'WELCOME TO OS/3 ICAM'
  94.0000           DC     X'0D'
  95.0000           DS     0H
+
  96.0000 MSGOT2    DC     X'001A'
  97.0000           DC     C'TYPE IN YOUR NAME PLEASE'
  98.0000           DC     X'0D1E'
  99.0000           DS     0H
 100.0000 MSGIN     DC     X'0000'
 101.0000           DC     CL5' '
 102.0000 TXTIN     DC     CL256' '
 103.0000 INETX     DC     X'FFFF'
 104.0000           DS     0H
 105.0000 MSGOUT    DC     X'010C'
 106.0000           DC     X'0D'
 107.0000           DC     C'THANK YOU '
 108.0000 TXTOUT    DC     CL256' '
 109.0000 OUTETX    DC     X'0D'
 110.0000 TXTCHARS  DC     X'0105'
 111.0000           END
```

Figure 2-31. Sample User Program that Uses Network NPR (Part 2 of 2)

Following are explanations of the statements and macroinstructions used in Figure 2-31.

Line	Explanation
2,3	Call DUST and DTFCP DSECTs
9 and 14	Establish DTFCP mapping registers
10-15	Map DTFCPs
19	Issue network request. Note that you can specify an ERRET= address only when you don't specify DUSTERR=INLINE in the CCA macroinstruction of the network definition.
23	Set up for complete message
24	Clear error field before each use
25	Send welcome message
31	Send type-in message
35	Clear error field before each GETCP
36	Set IRL for deferred GETCP
37	Set return address for deferred GETCP
38	Specify number of characters for GETCP
39	Issue GETCP
43	Our program yields
48-51	Message received; check for done message: *DONE* or *done*
55-57	Move text to output buffer and clear input buffer
58	Set output DTFCP for complete message
59	Echo complete message
60	Loop back to repeat welcome and type-in messages. Prepare for new input.
61	Release network - end of job
63,64, and 70	Release and rerequest line due to a line disconnect.

Line	Explanation
71,79	If a GETCP error, send message to operator, dump program, and terminate. (You must specify // OPTION DUMP in your job control to enable the snap dump.)
74,79	If a PUTCP error, send message to operator, dump program, and terminate
77,78	If no buffer available, send message to operator, dump program, and fall through to EOJ.
80	End of executable code.
84	Establish DTFCP for GETCP
85	Establish DTFCP for PUTCP
87–90	Define error messages
92–95	Set up welcome message: number of characters in message, message, and carriage return/line feed.
96–99	Set up type-in message: number of characters in message, message, and carriage return/line feed, SOE characters.
100–102	Input message buffer composed of: 1. a 2-byte length field you specify before you issue a GETCP. This value is updated with the number of bytes actually delivered by ICAM when the GETCP is honored; 2. a 5-character control character field composed of an SOE character and four DICE characters; this makes the text portion of a message fall into byte 7 (eighth byte) of the input message; and 3. a 256-byte text input area.
103	A 2-byte sentinel (hexadecimal FF) used to make it easier to find the end of the MSGIN work area and the beginning of the MSGOUT work area in a dump.
105–109	Output message buffer composed of: 1. Length field of output message; 2. Carriage return/line feed;

Line	Explanation
	3. Thank you message;
	4. 256-byte TXTOUT field; and
	5. A cursor return/line feed character.
110	A constant used in the length field of the GETCP

Now that we have described what our elementary program does, let's examine parts of it and see how it does what it does, with special emphasis on ICAM related functions. The following refers to a particular function in the program and to a statement or a group of statements in the program.

- Network Request

  When our program issues a network request (line 19), ICAM searches the symbiont to see if the network is present and is not busy. Also, because the NETREQ macroinstruction does not specify LNEREQ=(NO), all lines in this network are automatically activated. This is the default case. Note that line activation by your program is an option used only by dedicated network users. If you are using a global network, you must have the operator load the global user service task (GUST), which activates the requested global network and any lines the operator requests. You do not use a network request macroinstruction (NETREQ) in your global program because the network you want is already loaded when you execute your program. Instead, an NATTACH macroinstruction is used with similar parameters. See Figure 2-40 for an example of this same program written as a dynamic session global user program.

- Preparing to Answer the Phone

  Because network NPR specifies unattended answering for communications line LNE1, ICAM prepares itself to answer the telephone automatically when the network is loaded.

- Send Messages to the Terminal

  As our program continus to execute, it issues two PUTCP requests to ICAM to queue two messages for output (lines 25-31). Of course, the messages cannot be sent yet because the terminal user has probably not had a chance to dial the telephone number of the computer, so ICAM waits for a telephone call from the terminal. Remember, your program doesn't queue the output messages, ICAM does; ICAM takes the messages off the queue and sends them when it determines that conditions are right. This occurs when a telephone connection is made, and the terminal is operational and not busy. In this case, the terminal operator must dial the computer for any transmission to begin in either direction (input or output).

- Our Program Requests Input

  Our program requests a message from process file A (PRFA) by issuing a deferred GETCP. The deferred GETCP, set up on lines 36, 37, and 38 and issued on line 39 causes ICAM to access the low priority queue of process file A and deliver a message to our program's work area named MSGIN. When there is no message on the queue, ICAM suspends the GETCP until a message is queued. It returns control to our program at the next statement inline (in this case, a CYIELD statement). Our program is now yielded with nothing to do. (If our program had more terminals, related process files, or related input queues, we probably would have issued deferred GETCPs to all of them before yielding.) When a message arrives, the deferred GETCP that was suspended is honored, ICAM queues it to process file A, and our program receives control at the address specified on line 48, BINGO.

  If we had issued a nondeferred GETCP i.e., without IRL set in the DTFCP, the GETCP would be executed immediately; and if no message was available, control would be returned at the no-message-available address. Because a message could arrive any time, it would then be necessary to reissue the GETCP periodically to see if a message had arrived on the input queue. Deferred GETCPs are great time savers for your computer because they save looping through many GETCP requests to input queues, only to find that there is no message available.

- A Message is Delivered

  When our program receives control at the address specified in the deferred GETCP, it checks to see if the incoming message was *DONE* or *done* (uppercase or lowercase). If not, the message is moved to an output field and sent with a *THANK YOU* prefix. Notice that before any PUTCPs are issued, the TM#DSEG field of the DTFCP is set (TM#DSEG,TM#DHDR++TM#DTND). This tells ICAM that, for this PUTCP, an entire message is to be sent. Refer to the description of the DTFCP macroinstruction to see how to send partial messages, for example, one line at a time. The ++ is an assembler convention that indicates that the value represented by the two tags are to be ORed.

- Checking for Errors

  The error checking procedures in our program are rather primitive. We didn't make them more sophisticated because we didn't want to make this example too long or intricate. In most cases, we merely sent a message to the operator's console, dumped most of the program, and then terminated. However, we would like to make a few points related to error checking.

  1. Line disconnects

     When ICAM detects that any line has disconnected, it gives your program control at the address you specified in the NETERR operand of the NETREQ macroinstruction (line 19/line 63). Note that this is for dedicated networks only. At this time, ICAM places the name of the disconnected line in register 1, and the reason in byte 1 of register 0. You should always analyze these registers to determine why line disconnect occurred.

Notice that on lines 63 and 64 we issue a line release followed by a line request (LNEREL/LNEREQ). We did this so that ICAM answers the telephone when the next caller dials in following a disconnect. If you do not do this in your program, the line hangs and subsequent callers are not able to access the computer over this line. On line 70, we yield following the LNEREL/LNEREQ to regain synchronism with ICAM. For a more detailed description of how to analyze line disconnects, see 2.11.

2. GETCP/PUTCP errors

   If an error occurs because of an invalid GETCP/PUTCP request, your program receives control at the address you specified in the ERRET operand of the DTFCP macroinstruction for that GETCP or PUTCP (lines 71 and 74). In the case of a GETCP, you should always test TM#PIER in the TM#PIND field. If TM#PIER is set, it means that error settings are present in the error half-word TM#PERR. Also, you should remember to clear these error fields (TM#PIND and TM#PERR) before issuing a subsequent GETCP. ICAM provides similar error fields for PUTCP functions, and you should refer to the tables provided with the appropriate DTFCP macroinstruction descriptions in this user guide.

3. DSECT calls

   You will notice, at the beginning of the program, two DSECT calls (lines 2 and 3) are issued. This is done to make available to the assembler the addresses needed to calculate the locations of the various fields used in this program that are not explicitly defined in our program. Later, TM#PRCS and TM#DEST call those portions of the TM#DSECT DSECT needed to map the input (process file) and output (destination queue) DTFCPs. Notice on lines 10 and 15 that the addresses of the two DTFCPs are loaded into the registers specified in the USING statements. This lets the assembler calculate the addresses in your program for TM#DSEG (line 23) and TM#DERR (line 24).

   In addition to providing addresses to the assembler for mapping purposes, the DSECTs provide names (tags) to represent hexadecimal values. These hexadecimal values are used to set values in the DTFCPs and to enable your program to test for error conditions without needing to know the exact error values. For example, in line 16, our program tells the assembler to take the value represented by TM#DHDR and combine it (OR it) with the value represented by the TM#DTND and place the result in a field named TM#DSEG. When the program is executed, this tells ICAM to send a complete message. By using the DSECT tags you make it easy to update your program when you move to a new release. All you need to do is reassemble your program and the assembler changes all of the values automatically. As a practical matter, you rarely need to know the actual value represented by a tag, as long as the tag represents the function you want performed.

   ICAM DSECTs that you can use are summarized in 2.18.

We executed user program USERV9 by means of the following command:

```
// EXEC USERV9
```

and the result was the traffic displayed on the UTS 400 shown in Figure 2-32.

```
WELCOME TO △OS/3 ICAM          (welcome statement)
TYPE IN YOUR NAME PLEASE       (type-in statement)
JOHN Q PUBLIC                  (operator response)
THANK YOU JOHN Q PUBLIC        (program response)

WELCOME TO △OS/3 ICAM          (welcome statement)
TYPE IN YOUR NAME PLEASE       (type-in statement)
DONE                           (operator response)
```

NOTE:

Program terminates because *DONE* was entered.

Figure 2-32. Execution of Program USERV9

2.23.3. An ICAM Environment Supporting Three Communications Lines

The following example shows an ICAM network that supports three communications lines. The first line supports four SPERRY UNIVAC UTS 400 terminals, the second line supports a Teletype Corporation Model 33 teletypewriter unit as a single station, and the third line supports a SPERRY UNIVAC 1004 system. Figure 2-33 illustrates the communications system; Figure 2-34 shows the network definition coding to support it. The user program shown in Figure 2-35 only deals with the teletypewriter unit. It echos any message received from the teletypewriter device back to it. For example, ICAM queues any message received from teletypewriter TRM5 to process file PRF2. The user program:

- obtains the message from the process file by issuing a GETCP;
- moves the message just received into an output work area;
- sets up an output DTFCP; and
- sends the message back to the teletypewriter unit using a PUTCP exactly as it was received.

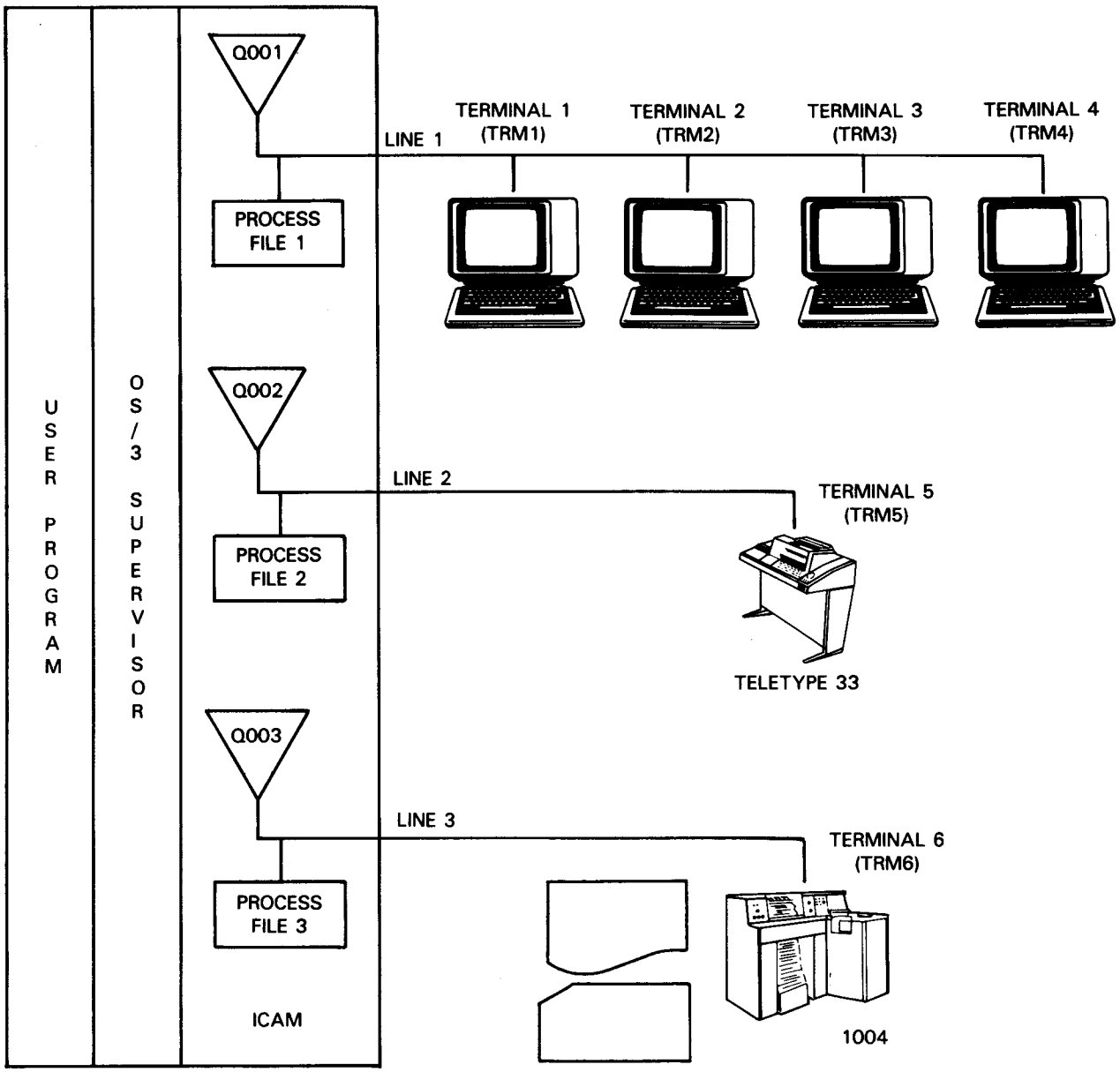


Figure 2-33. Graphic Representation of ICAM Network

```
1         10        16                                          72
NET1      CCA       PASSWORD=AUTH3726                           X
                    TYPE=(STDMCP),
          BUFFERS 20,100,ARP=25
LNE1      LINE      CALL=6461062,                               X
                    DEVICE=(UNISCOPE),                          X
                    TYPE=(2400,UNAT,SWCH,SYNC),                 X
                    INPUT=(PRF1)                                X
                    ID=4,                                       X
                    LOW=MAIN
TRM1      TERM      ADDR=(29,53),                               X
                    FEATURES=(U400,1024),                       X
                    PINTV=50
TRM2      TERM      ADDR=(29,54),                               X
                    FEATURES=(U400,1024),                       X
                    PINTV=50
TRM3      TERM      ADDR=(28,51),                               X
                    FEATURES=(U400,1024),                       X
                    PINTV=50
TRM4      TERM      ADDR=(28,52),                               X
                    FEATURES=(U400,1024),                       X
                    PINTV=50
LNE2      LINE      CALL=6460262,                               X
                    DEVICE=(TTY,33),                            X
                    TYPE=(100,SWCH),                            X
                    INPUT=(PRF2)                                X
                    ID=6,                                       X
                    LOW=MAIN
TRM5      TERM      ANSWER=(8,C,BLUEBELL),                      X
                    FEATURES=(TTY)
LNE3      LINE      CALL=6462766,                               X
                    DEVICE=(1004),                              X
                    TYPE=(2400,SWCH,SYNC),                      X
                    INPUT=(PRF3)                                X
                    LOW=MAIN
TRM6      TERM      ANSWER=(8,C,BLUEBELL),                      X
                    FEATURES=(1004)
PRF1      PRCS      LOW=MAIN
PRF2      PRCS      LOW=MAIN
PRF3      PRCS      LOW=MAIN
          ENDCCA
```

Figure 2-34. Network Definition for ICAM Network

```
1         10    16                                                    72
MCOTTY    START Ø
          TN#DSECT DUST                        USER DSECTS
          TM#DSECT PRCS
          TM#DSECT DEST
BEGIN     BALR  1Ø,Ø                           INITIALIZE THE RELATIVE STARTING
          USING *,1Ø                             ADDRESS OF THE PROGRAM
          USING TM#PRCS,2                      COVER REGISTERS
          USING TM#DEST,3
          USING TQ#DSCTS,4
          LA1,MSG1
          OPR   MSG1,MSG1L                     OPERATOR MESSAGE
          LA    2,PRF2                         MAP INPUT FILE TABLE
          LA    3,Q002                         MAP OUTPUT FILE TABLE
          NETREQ NET1,PASSWORD=AUTH3726,       REQUEST THE NETWORK              X
             ERRET=ERRORS*
          XC    TM#PERR,TM#PERR                CLEAR ERROR INDICATOR
GETIT     GETCP PRF2,WORKIN                    GET INPUT MESSAGE
          CLC   WORKIN+7(7),=C'RELEASE'        COMPARE FOR LAST MESSAGE
          BE    ENDJOB
          MVC   WORKOUT,WORKIN                 MOVE INPUT BUFFER CONTENTS
                                                   TO OUTPUT
          OI    TM#DSEG,TM#DHDR++TM#DIND       SET FOR COMPLETE MESSAGE
          XC    TM#DERR,TM#DERR                CLEAR ERROR INDICATOR
          PUTCP Q002,WORKOUT                   SEND OUTPUT MESSAGE
          B     GETIT
ENDJOB    LA    1,MSG5
          OPR   MSG5,MSG5L
          LNEREL LNE1                          RELEASE THE LINE
          NETREL NET1                          RELEASE THE NETWORK
          EOJ
ERRORS    LA    1,MSG2
          OPR   MSG2,MSG2L                     NETWORK REQUEST OR DUST
          SNAP  BEGIN,MSG5L                      ERROR MESSAGE
NOGET     LA    1,MSG3
          OPR   MSG3,MSG3L                     INPUT ERROR MESSAGE
          SNAP  BEGIN,MSG5L
NOPUT     LA    1,MSG4
          OPR   MSG4,MSG4L                     OUTPUT ERROR MESSAGE
          SNAP  BEGIN,MSG5L
          DS    ØH'
WORKIN    DC    CL1ØØ'△'                       INPUT BUFFER
          DS    ØH
WORKOUT   DC    CL1ØØ'△'                       OUTPUT BUFFER
          DS    ØF
PRF2      DTFCP TYPE=GT,ERRET=NOGET            INPUT FILE DTFCP
          DS    ØF
Q002      DTFCP TYPE=PT,DEST=(T,TRM5),ERRET=NOPUT OUTPUT FILE DTFCP
MSG1      DC    C'THIS IS MCOTTY'
MSG1L     EQU   L'MSG1
```

Figure 2-35. User Program for ICAM Network (Part 1 of 2)

```
1         10    16                                                    72
MSG2      DC    C'NETWORK OR LINE REQUEST REJECTED--JOB WILL TERMINATE'
MSG2L     EQU   L'MSG2'
MSG3      DC    C'GET REJECTED-JOB WILL TERMINATE'
MSG3L     EQU   L'MSG3'
MSG4      DC    C'PUT REJECTED-JOB WILL TERMINATE'
MSG4L     EQU   L'MSG4'
MSG5      DC    C'END OF MCOTTY'
MSG5L     EQU   L'MSG5'
          END
```

* You can specify an ERRET= address only when you don't specify DUSTERR= INLINE in the CCA macroinstruction.

Figure 2-35. User Program for ICAM Network (Part 2 of 2)

2.23.4. How to Write a Program to Use ICAM in a Global Network Environment

Programs that access global networks are functionally similar to those that access dedicated networks. You use acquire/release macros to attach to specific networks and open specific sessions, as well as to close sessions and detach networks.

Once you establish a session, the send/receive macroinstructions for message transfer in global networks are identical to those in dedicated networks. For the standard interface (STDMCP), this means the GETCP/PUTCP macroinstructions.

In the global network, you must initialize GUST before you can attach (NATTACH) to the network. (See the OS/3 ICAM network definition and operations user guide, UP-8947 (current version).) The reason for the initialization is that GUST (not your program) owns the communication lines in the global network. Also, GUST attempts to keep the lines active, depending on the type of connection procedure, for dialing and data flow. Because GUST controls the communication lines, the LNEREQ and LNEREL macroinstructions in a program are interpreted as no-ops in a global network.

The following is an example of a global network in a single node (one computer). In this example, user program 1 (CUP1) receives input from terminal 1 (TRM1) and terminal 2 (TRM2) sends it to another user program (CUP2). CUP2 processes the terminal data and replies to CUP1. To simplify the example, we divided it into:

- Figure 2-36 showing the generation of the network;
- Figure 2-37 showing the flowchart of CUP1; and
- Figure 2-38 showing the program code for CUP1.

The error logic for CUP1 and the program code for CUP2 are not shown.

Looking at Figure 2-36, you will notice that the global network generation for a single node system is similar to a dedicated network system with the exception of the LOCAP and SESSION macroinstructions. The LOCAP macroinstruction defines a program similar to a process file enabling the messages to be queued directly to it. The SESSION macroinstruction defines the names of the end users communicating with each other.

```
      1         10    16                                                  72
      COMMCT
1.    GNSN      CCA   TYPE=(GBL,,S),FEATURES=(OPCOM)
                BUFFERS 40,64,9,ARP=35,STAT=YES
2.    CUP1      LOCAP TYPE=(STDMCP),LOW=FILEA,MEDIUM=FILEA,HIGH=FILEA
3.    CUP2      LOCAP TYPE=(STDMCP),LOW=FILEA,MEDIUM=FILEA,HIGH=FILEA
4.    LNE1      LINE  DEVICE=(UNISCOPE),TYPE=(2000,SWCH,SYNC),ID=12
5.    TRM1      TERM  ADDR=(28,51),FEATURES=(U200,2048),                  X
                      LOW=FILEA,MEDIUM=FILEA,HIGH=FILEA,                 X
                      INPUT=(YES,,FILEA)
      TRM2      TERM  ADDR=(28,52),FEATURES=(U200,2048),                  X
                      LOW=FILEA,MEDIUM=FILEA,HIGH=FILEA,                 X
                      INPUT=(YES,,FILEA)
6.    TRM3      TERM  ADDR=(29,53),FEATURES=(U200,2048),                  X
                      LOW=FILEA,                                         X
                      INPUT=(TRM4)
      TRM4      TERM  ADDR=(29,54),FEATURES=(U200,2048),                  X
                      LOW=FILEA,                                         X
                      INPUT=(TRM3)
7.              SESSION EU1=(CUP1),EU2=(CUP2)
                SESSION EU1=(CUP1),EU2=(TRM1)
                SESSION EU1=(CUP1),EU2=(TRM2)
                SESSION EU1=(TRM3),EU2=(TRM4)
8.    FILEA     DISCFILE FILEDIV=15
                ENDCCA
                MCP
                MCPVOL=REL071
                MCPNAME=C1
                      CACH=(12,GNSN,1)
      END
      // FIN
```

NOTES:

1. Designates this as a global network with the node name of S.

2. Defines a LOCAP named CUP1 with three disk queues.

3. Defines a LOCAP named CUP2 with three disk queues.

4. Defines a local switched synchronous line operating at 2000 baud and using UNISCOPE terminals.

5. Defines a UNISCOPE terminal (U200) on LINE1 with a rid and sid address of (28,51) with three disk output queues and a disk input queue (FILEA).

6. Defines another UNISCOPE 200 terminal on LINE1 with a rid and sid address of (29,53) and a low priority output queue on disk; input from terminal 3 is sent to the output queues for terminal 4.

7. Defines the session as follows:

   CUP1 and CUP2
   CUP1 and TRM1
   CUP1 and TRM2
   TRM3 and TRM4

8. Defines the disk file for the queues.

Figure 2-36. SYSGEN of a Typical Global Network

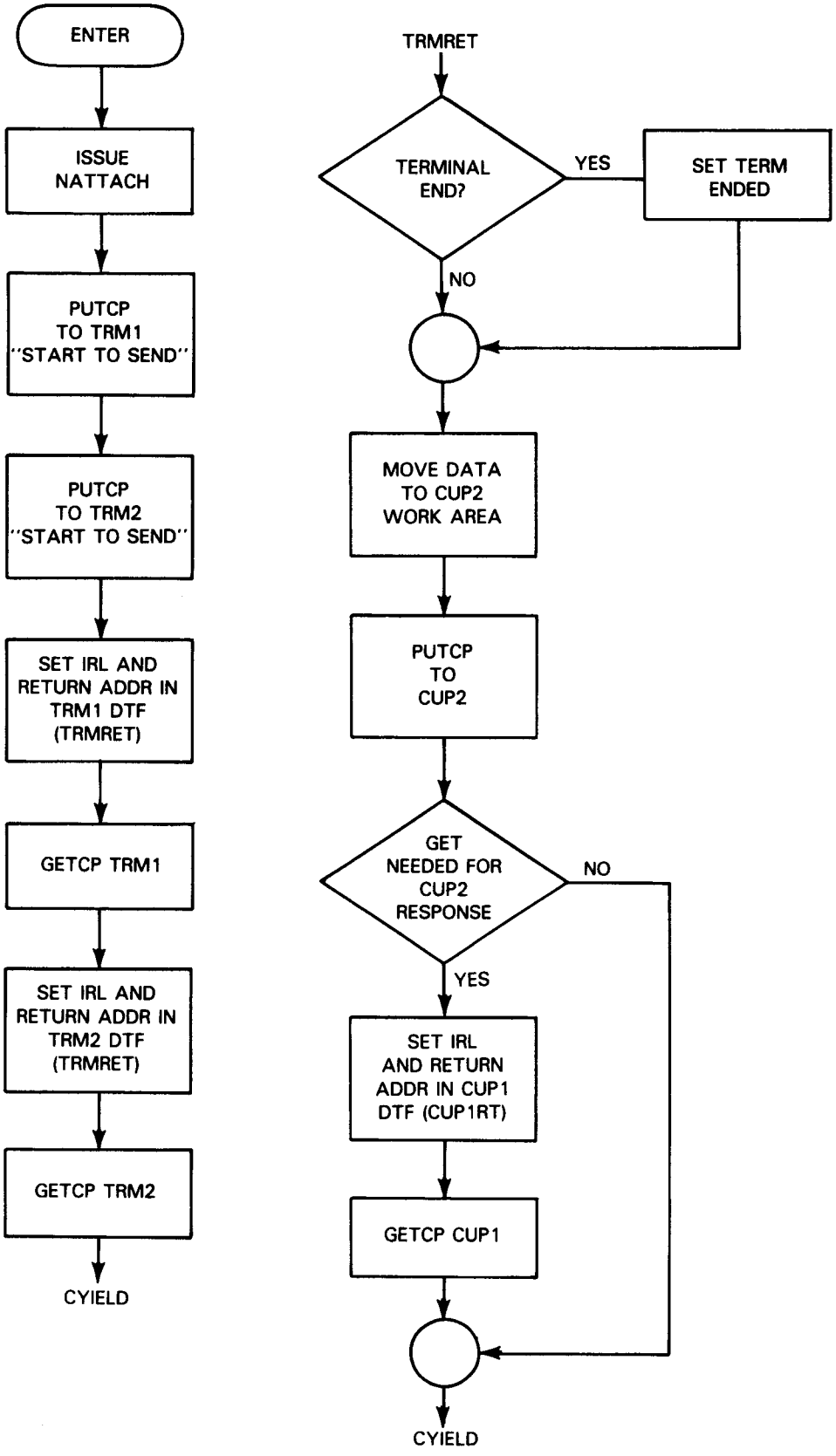


Figure 2-37. Flowchart of Global User Program (Part 1 of 2)

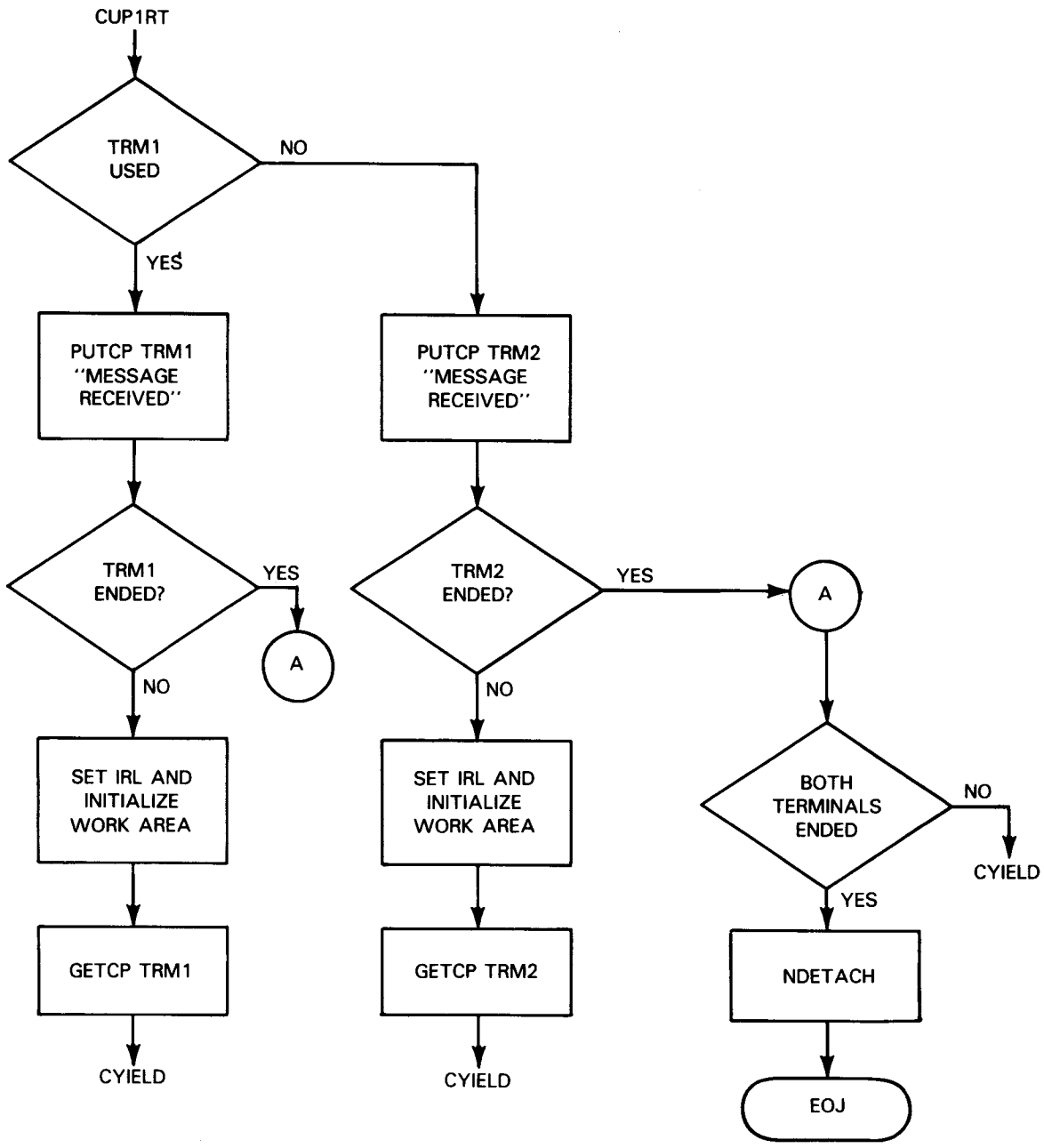


Figure 2-37. Flowchart of Global User Program (Part 2 of 2)

```
@P
   1.0000 USERA    START 0
   2.0000          TM#OSECT PRCS,DEST
   3.0000          BALR  R8,0                      ESTABLISH COVER
   4.0000          USING *,R8
   5.0000          NATTACH GNSN,APPS=CUP1,ERRET=ERROR ATTACH TO NETWORK
   6.0000          USING TM#PRCS,R1                COVER THE DTF FOR GETCP
   7.0000          PUTCP T1OT,M1START              PUT START MESSAGE TO TRM1
   8.0000          PUTCP T2OT,M1START              PUT START MESSAGE TO TRM2
   9.0000          LA    R1,TRM1
  10.0000          MVI   TM#PIND,TM#PIRL           SET IRL INTO THE DTF
  11.0000          MVC   TM#PCMPA,=A(TRMRET)       ESTABLISH IRL RETURN IN DTF
  12.0000          GETCP TRM1,T1WORK               GET A MESSAGE FROM TRM1
  13.0000          LA    R1,TRM2
  14.0000          MVI   TM#PIND,TM#PIRL           SET IRL INTO THE DTF
  15.0000          MVC   TM#PCMPA,=A(TRMRET)       ESTABLISH IRL RETURN IN DTF
  16.0000          GETCP TRM2,T2WORK               GET A MESSAGE FROM TRM2
  17.0000
  18.0000          B     CYIELD                    GOTO THE COMMUNICATIONS YIELD
  19.0000 *
+
  20.0000 *
  21.0000 TRMRET   EQU   *                         TERMINAL INPUT ENTRY POINT
  22.0000          LR    R2,R0                     GET THE ADDRESS OF WORK AREA
  23.0000          CLC   EOTRAN,7(R2)              'END DATA' RECEIVED?
  24.0000          BNE   RESPOND                   NO - GOTO RESPOND
  25.0000          C     R1,=A(TRM1)               TRM1 ENDED?
  26.0000          BNE   TRM2EOT                   NO - GOTO TRM2EOT
  27.0000          MVI   ENDTRM1,X'FF'             SET TRM1 ENDED
  28.0000          B     RESPOND                   GOTO RESPOND
  29.0000 TRM2EOT  EQU   *                         'END DATA' FROM TRM2
  30.0000          MVI   ENDTRM2,X'FF'             SET TRM2 ENDED
  31.0000 RESPOND  EQU   *                         RESPOND TO INPUT
  32.0000          MVC   C2WORK(40),0(R2)          MOVE DATA TO CUP2 OTPT BUFFER
  33.0000          PUTCP C2OT,C2WORK               PUT MESSAGE TO CUP2
  34.0000          CLI   CUP2GET,X'FF'             GET OUTSTANDING TO CUP2?
  35.0000          BE    CYIELD                    YES - GOTO CYIELD
  36.0000          LA    R1,CUP1
  37.0000          MVI   TM#PIND,TM#PIRL           SET IRL INTO THE DTF
  38.0000          MVC   C1WORK,=H'1'              SET BUFFER LENGTH
+
  39.0000          MVI   C1TEXT,C' '               CLEAR THE BUFFER
  40.0000          MVC   TM#PCMPA,=A(CUP1RT)       ESTABLISH IRL RETURN IN DTF
  41.0000          GETCP CUP1,C1WORK               GET A MESSAGE FROM CUP2
  42.0000          MVI   CUP2GET,X'FF'             SET GET OUTSTANDING TO CUP2
  43.0000          B     CYIELD                    GOTO COMMUNICATIONS YIELD
  44.0000 *
  45.0000 *
  46.0000 CUP1RT   EQU   *                         MESSAGE RECEIVED FROM CUP2
  47.0000          MVI   CUP2GET,0                 CLEAR CUP2 GET OUTSTANDING
  48.0000          CLI   C1TEXT,C'1'               TRM1?
  49.0000          BNE   PUTTR2                    NO - GOTO PUTTR2
  50.0000          PUTCP T1OT,M2START              RESPOND TO TRM1
```

Figure 2-38. Sample Global User Program (CUP1) (Part 1 of 3)

```
  51.0000           CLI   ENDTRM1,X'FF'                TRM1 ENDED?
  52.0000           BE    CHEKYLD                      GOTO CHEKYLD
  53.0000           LA    R1,TRM1
  54.0000           MVI   TM@PIND,TM@PIRL              SET IRL INTO THE DTF
  55.0000           MVC   T1WORK,=H'38'                SET BUFFER LENGTH
  56.0000           MVI   T1TEXT,C' '                  CLEAR THE BUFFER
  57.0000           MVC   T1TEXT+1(37),T1TEXT
+
  58.0000           GETCP TRM1,T1WORK                  GET A MESSAGE FROM TRM1
  59.0000           B     CYIELD                       GOTO COMMUNICATIONS YIELD
  60.0000 *
  61.0000 *
  62.0000 PUTTR2    EQU   *                            TRM2
  63.0000           PUTCP T2OT,M2START                 RESPOND TO TRM2
  64.0000           CLI   ENDTRM2,X'FF'                TRM2 ENDED?
  65.0000           BE    CHEKYLD                      YES - GOTO CHEKYLD
  66.0000           LA    R1,TRM2
  67.0000           MVI   TM@PIND,TM@PIRL              SET IRL INTO THE DTF
  68.0000           MVC   T2WORK,=H'38'                SET BUFFER LENGTH
  69.0000           MVI   T2TEXT,C' '                  CLEAR THE BUFFER
  70.0000           MVC   T2TEXT+1(37),T2TEXT
  71.0000           GETCP TRM2,T2WORK                  GET A MESSAGE FROM TRM2
  72.0000           B     CYIELD                       GOTO COMMUNICATIONS YIELD
  73.0000 *
  73.1000
  74.0000 *
+
  75.0000 CHEKYLD   EQU   *                  YLD TASK IF TERMINAL'S ACTIVE
  76.0000           CLI   ENDTRM1,X'FF'                TRM1 ENDED?
  77.0000           BNE   CYIELD                       NO - CYIELD
  78.0000           CLI   ENDTRM2,X'FF'                TRM2 ENDED?
  79.0000           BE    NDETACH                      YES - GOTO NDETACH
  80.0000 *
  81.0000 *
  82.0000 CYIELD    EQU   *
  83.0000           CYIELD                             YLD TASK FOR GETCP COMPLETION
  84.0000 NDETACH   EQU   *                            DETACH FROM NETWORK
  85.0000           NDETACH GNSN                      DETACH
  86.0000           EOJ
  87.0000 *
  88.0000 *
  89.0000 ERROR     EQU   *                            ERROR EXIT
  90.0000           ERROR LOGIC NOT SHOWN
  91.0000           B     NDETACH                      GOTO NDETACH
  92.0000 *
  93.0000 *
+
  94.0000           CNOP  0,4
  95.0000 T1OT      DTFCP TYPE=PT,DEST=(T,TRM1),LEVEL=LOW,ERRET=ERROR
  96.0000 T2OT      DTFCP TYPE=PT,DEST=(T,TRM2),LEVEL=LOW,ERRET=ERROR
  97.0000 C2OT      DTFCP TYPE=PT,DEST=(P,CUP2),LEVEL=MEDIUM,ERRET=ERROR
  98.0000 TRM1      DTFCP TYPE=GT,ERRET=ERROR
```

Figure 2-38. Sample Global User Program (CUP1) (Part 2 of 3)

```
 99.0000 TRM2     DTFCP TYPE=GT,ERRET=ERROR
100.0000 CUP1     DTFCP TYPE=GT,LEVEL=MEDIUM,ERRET=ERROR
101.0000 *
102.0000 *
103.0000 M1START  DC    H'16'                    DATA LENGTH
104.0000          DC    X'0D'                    LINE FEED
105.0000          DC    C'START TO SEND'
106.0000          DC    X'0D'                    LINE FEED
107.0000          DC    X'1E'                    SOE CHARACTER
108.0000 M2START  DC    H'19'                    DATA LENGTH
109.0000          DC    X'0D'                    LINE FEED
110.0000          DC    C'MESSAGE RECEIVED'
111.0000          DC    X'0D'                    LINE FEED
112.0000          DC    X'1E'                    SOE CHARACTER
+
113.0000 EOTRAN   DC    C'END DATA'
114.0000 ENDTRM1  DC    X'0'                     X'FF'-->END INPUT FROM TRM1
115.0000 ENDTRM2  DC    X'0'                     X'FF'-->END INPUT FROM TRM2
116.0000 CUP2GET  DC    X'0'                     X'FF'-->GETCP TO CUP2
117.0000 T1WORK   DC    H'38'                    INPUT BUFFER FOR TRM1
118.0000 T1TEXT   DC    CL38' '
119.0000 T2WORK   DC    H'38'                    INPUT BUFFER FOR TRM2
120.0000 T2TEXT   DC    CL38' '
121.0000 C1WORK   DC    H'1'                     INPUT BUFFER FOR CUP1
122.0000 C1TEXT   DC    CL1' '
123.0000 C2WORK   DC    H'38'                    OUTPUT BUFFER TO CUP2
124.0000 C2TEXT   DC    CL38' '
125.0000          END
```

NOTE:

You can specify an ERRET= address only when you don't specify DUSTERR=INLINE in the CCA macroinstruction.

Figure 2-38. Sample Global User Program (CUP1) (Part 3 of 3)

2.23.4.1. Programming Considerations for Global Networks

Note the following considerations for use of global networks:

- Static sessions between end users are permanently assigned by the network definition process.
- You must use GUST (executed by the program ML$$GI) to use global networks.
- The GUST has no configuration levels.
- If your program detaches from the global network without letting active message flow complete for all sessions, or if either ICAM or GUST terminates normally or abnormally with active users, GUST removes the messages without notifying any of the session end users.
- A message can be duplicated at the recipient node if the trunk is reestablished after a trunk-down condition when using the GLONET X.25 interface.
- A global user program can define up to three special queues through the HIGH, MEDIUM, and LOW operands in a LOCAP macro. These queues belong only to the program named in the label of the LOCAP macro, and only that program can access messages placed on them. To access the queues, a program uses a GETCP and references its name in the related DTFCP for the Get request. To place messages on these queues, any other end user uses a PUTCP in the same manner as queueing a message to a process file, and addresses the user program.

  Only ICAM can access the queues to transfer queued messages to a user program residing in a remote node, defined by the REMOTE operand in the LOCAP macro. In this case, the queues become destination queues for the remote user program.
- Through the HIGH, MEDIUM, and LOW operands of the TERM macro, you can create up to three priority queues for messages destined to each terminal. You can also create a single input queue for all input messages from the terminal through the INPUT=YES operand in the TERM macro. Your program could access the input queue through the label specified on the TERM macro in the related DTFCP for the Get request.

  In addition, you can direct the terminal input to a local end user through the INPUTend-user-name in the TERM macro. For example, if all terminal input is going to another terminal called TRM2, the label of the destination terminal in the INPUT operand of the source terminal's TERM macro is INPUT=TRM2.

  Note that, when you issue a SESCON macroinstruction to establish a dynamic session, you can dynamically change (or define) the destination for input from a terminal. In other words, the SESCON macroinstruction INQNAME operand overrides any destination you specified in a TERM macroinstruction when you generated your ICAM network.

- Process file queues temporarily store messages destined for a user program. Any program can issue a Put request to place messages on a process file or issue a Get request to access a message existing on a process file. The low, medium, and high message queues are both destination queues (known as the Put side) and input queues (known as the Get side).

  If a message is directed to a process file located in a remote node, the message is stored on a local process file with the same name as the remote process file. The MCP accesses the Get side of this process file only to transfer a queued message over the virtual link (VLINE) to the destination process file. The final process file is in the remote node and any remote user program can access the message. This design prevents local programs from accessing process files and removing messages destined for remote nodes.

2.24. DYNAMIC SESSIONS

The exchange of information between two communications end users in a session requires that a path be created through the communications system. In the ICAM global network described in 2.23.4, the path is created at system generation time for specifically paired end users. This type of session is called a static session.

In a dynamic session, certain end users can dynamically establish a session with any one of a designated group of end users. ICAM can dynamically acquire and construct required tables, construct the path between paired end users, and release tables when the session is completed.

When using OS/3 with the single line communications adapter or distributed communications processors, a dynamic session can be established from:

- an ICAM terminal to a user program, terminal, or process file; or
- a user program to an ICAM terminal, process file, or another user program.

NOTE:

Sessions between user programs in the same computer node do not need the communications adapter.

See Figure 2-40 for a working example of a dynamic session user program.

2.24.1. Session Establishment

2.24.1.1. Establishing a Session from a Terminal

You can request a dynamic session with another end user from a terminal by typing in the following network sign-on command:

```
$$SON xxxxyyyy
```

where:

xxxx
: Is the logical name of your initiating terminal.

yyyy
: Is the logical name of the called end user (either another terminal, a user program, or a process file).

The logical names are specified during network definition.

You disestablish a session from a terminal with the $$SOFF sign-off command.

A number of standard messages are provided by ICAM to inform the terminal operator of communications path conditions and thereby provide guidance in terminal operations. The messages listed in Table 2-46 are displayed on the terminal during session operations to indicate conditions as described in the table.

Table 2-46. Terminal Messages for a Dynamic Session

Message Number	Message	Meaning
MSG1	SPERRY-UNIVAC DCA NETWORK, LEVEL x.x, NODE yyyy	Indicates current ICAM release level and node.
MSG2	SESSION PATH OPEN	Indicates that terminal user can exchange normal data from this point on.
MSG3	SESSION PATH CLOSED	Response to CUP request to close session or session is rejected for CUP or terminal.
MSG4	SESSION PATH ABORTED	Response to terminal request to close session or indicates that session is aborted by the CUP.
MSG5	$$SOFF	Indicates that session is deleted from the ICAM environment.
MSG6	INVALID $$ COMMAND	Indicates an invalid command.

NOTE:

Message numbers are not displayed. They are only used for convenience in reference.

2.24.1.2. Establishing a Session from a User Program

You can establish a session from your communications program by means of a SESCON macroinstruction. By properly defining the FUNCT parameter of the SESCON macroinstruction, you can perform the following functions:

- Establish an end user session (FUNCT=OPEN)
- Accept a request from another end user to open a session (FUNCT=OPNACC)
- Reject a request to open a session from another end user (FUNCT=OPNREJ)
- Disestablish a session in progress in an orderly manner (FUNCT=CLOSE)
- Abnormally terminate a session (FUNCT=ABORT)
- Confirm a request to close a session from another end user and disestablish the session in an orderly manner (FUNCT=CLSCNF)

The SESCON macroinstruction is fully described in 2.22.1.7.

2.24.2. Session Description

2.24.2.1. Session Open from a Terminal

When a terminal operator signs on with the $$SON, MSG1 (Table 2-46) is displayed on the terminal defining the ICAM release level and node.

ICAM, on receipt of the open request, validates the request and acquires the necessary resources to create the session path. If ICAM fails to establish a path, it rejects the open request for a session, and sends MSG3 to the terminal.

If ICAM succeeds in creating the session path for the terminal, an OPEN datagram is sent to the called end user if it is a user program. The user program responds to the request by either accepting or rejecting it via the SESCON macroinstruction (FUNCT=OPNACC or FUNCT=OPNREJ).

If the user program rejects the open request, the terminal is signed off the network with MSG3 and MSG5. If the user program accepts the open request, MSG2 is displayed indicating that the session path is open and normal data can be exchanged between the paired end users.

When the called end user is another terminal and there are no sessions in progress with this terminal, MSG1 and MSG2 are displayed on the called end user terminal. MSG2 now follows MSG1 on the display of the calling terminal. If, however, ICAM is unable to create this session, MSG3 follows MSG1 on the display of the calling terminal.

When the called end user is a process file, resources are available to establish this session, and there is not more than one other session in progress with this process file, then MSG2 follows MSG1 on the display of the calling terminal. This indicates that message transfer may now begin. Otherwise, MSG3 is displayed indicating that the session path was not created.

2.24.2.2. Session Open from a User Program

The user program requests a dynamic session via the SESCON macroinstruction (FUNCT=OPEN). The request is validated by ICAM and the resources to create the session path are obtained. If the request is invalid or resources are not available, the session request by the user program is rejected by ICAM and an OPNREJ datagram is sent to the user program.

If ICAM accepts the session and the called end user is a terminal, MSG1 and MSG2 are displayed on the called terminal and an open accept datagram (OPNACC) is returned to the calling user program.

When the called end user is another user program and the session is accepted by ICAM (i.e., resources are available for this valid request), a FUNCT=OPEN datagram is sent to the called user program. The called user program must then either accept or reject the open request via a SESCON macroinstruction with FUNCT=OPNACC or OPNREJ. This information is then transferred to the calling user program via datagrams that indicate the request to open is either accepted or rejected. If the request is accepted by the called user program, normal data transfer begins between these paired end users.

If the user program requests a dynamic session to a process file, ICAM validates the SVC and, if valid, responds to the user program with an OPNACC datagram. If the SVC is invalid, resources are not available, or two sessions are already in progress with the called process file, an OPNREJ datagram is sent to the calling user program.

2.24.2.3. Session Close from a Terminal

After a session is opened and normal data is flowing, either of the paired end users can close the session. The terminal operator signs off with $$SOFF.

The $$SOFF is interpreted as a request to abort the session. When ICAM receives the request, the session is disestablished and MSG3 is displayed on the requesting terminal screen. An ABORT control datagram is sent to the paired end user program as notification of session termination. No further action is required of the user program to end the session. When a session is aborted, any message in transit to the terminal is the last. If the paired end user is another terminal, MSG4 is displayed on the called end user screen before the terminals are signed off.

2.24.2.4. Session Close from a User Program

When the user program issues a close request via the FUNCT=CLOSE parameter of the SESCON macroinstruction, ICAM closes the output path to the called end user. The process following this depends on whether the called end user is another user program, a terminal, or a process file as follows:

- User Program

  If the called end user is another user program, ICAM sends the user program a CLOSE datagram and waits for a response. The called user program sends a close confirm SESCON macroinstruction after it has sent all the data it wants to send to the calling end user. When ICAM receives the close confirmation from the called end user, the session is disestablished and a datagram indicating the close confirmation is sent to the calling end user, thus signifying the end of the session. When the called end user program is returned control from the FUNCT=CLSCNF parameter of the SESCON macroinstruction, the called user program is disestablished from the session.

- Terminal

  If the called end user is a terminal, MSG3 and MSG5 are displayed on the terminal after all the output data is delivered. A close confirm control datagram is sent to the calling end user program to indicate the end of the session.

- Process File

  If the called end user is a process file, a close confirm control datagram is sent to the calling end user program to indicate the end of the session.

  NOTE:

  When you direct messages from your terminal to another end user such as a process file rather than the terminal input queue, it is possible for you to receive responses to the messages after a dynamic session is disestablished and SESSION PATH CLOSED is displayed on your screen.

  When you sign off, the terminal input queue, the destination queues, and terminal control table are all cleared. Responses to messages on the input queue are not possible after session disestablishment except if a session is disestablished after a user program gets a terminal message, but before it can put a response to the terminal.

  If terminal messages are directed to the process file instead of the terminal input queue, they are cataloged with messages from other end users and are not cleared when the session is disestablished. Therefore, the user program can continue to get the messages, which are related to the session, from the process file after the session is terminated.

Therefore, to reduce the possibility of receiving messages at a terminal after the session is disestablished, specify INPUT=YES in the TERM macroinstruction to hold terminal input messages on the terminal input queue.

2.24.2.5. Saving User Program Messages

Whenever an end user receives a close confirm or an abort request message, ICAM automatically deallocates the session and all resources because the end user cannot acknowledge receipt of the message. This can cause the loss of the message on the input queue of the terminal table unless you take certain precautions.

ICAM does not have the means to have the user program process the input queue messages before the close confirm or abort request is sent to the user program via a datagram. However, before sending the datagram to the user program, ICAM must make sure that the queues associated with the requesting end user are cleared, because the terminal is now free to sign on to another session.

Therefore, to preserve all messages destined for the user program, the user program must not use the input queue associated with the reused terminal table (generated by INPUT=YES on the TERM macroinstruction). The user program can direct all input to a LOCAP or to another process file using the INQNAME parameter of the SESCON macroinstruction.

2.24.3. Control Datagrams

A program issues the SESCON macroinstruction to establish or disestablish a session either on its own initiative or in response to such a request from another end user. Special types of control datagrams are used to inform the user program of such requests or responses from the paired end user. User programs initiating dynamic session establishment must specify the TYPE=INPUT parameter of the GAWAKE macroinstruction after attaching to ICAM via the NATTACH macroinstruction.

The input buffer size specified for the datagram must not be smaller than the largest session control datagram that it may receive (see the BUFLTH operand, 2.22.2.3). The minimum buffer size is defined by equate xxKLSZ in the DSECT that covers the datagram. However, it is suggested that you allow for a larger buffer size than that defined by the equate to avoid reassembling your program with each release. The size of the control datagram could change with a new software release as more information is passed to your program.

The control datagram is identified by the $$ characters of the text received by the user program and the type of datagram in the half word following the $$ field.

Control datagrams are used by ICAM to inform a user program of session establishment requests or responses initiated by another end user. Datagrams are transferred to a user program in a work area supplied by the user program. To receive the datagrams, a user program must first issue a GAWAKE macroinstruction with the TYPE=INPUT parameter. Session establishment should then proceed as follows:

- From a terminal:

- ICAM schedules a datagram (OPNREQ) to your program.

- Your program issues a CYIELD and receives and processes the datagram.

- Your program issues a SESCON (FUNC=OPNACC or OPNREJ) and receives a SESCON completion.

- Additional end user-initiated sessions, if any, are established by repeating steps 1 through 3.

- From a user program:

- Your program issues a SESCON with FUNC=OPNREQ and receives a SESCON completion.

- ICAM processes the SESCON and schedules an OPNACC or OPNREJ datagram for your program.

- Your program issues a CYIELD and receives and processes the datagram.

- Your program establishes additional program-initiated sessions, if any, by repeating steps 1 through 3.

The GAWAKE macroinstruction has parameters for specifying the datagram work area, work area length, and an address to which control is returned when a datagram is made available to the user program.

The GAWAKE TYPE=INPUT parameter notifies ICAM that the user program is ready to receive any datagrams directed to it. Thereafter, when a datagram is available, it is transferred to the specified work area and control is returned to the specified address whenever the user program yields control to ICAM.

The formats of the various types of control datagrams and their parameter lists are shown in Figure 2-39 and Table 2-47. The datagrams are used as indicated in 2.23.4.1 and each parameter is explained in 2.24.3.2.

OPEN DATAGRAM

$	$	KQTYP (function code)	← KQTY1 (KOPEN)
(from) K1CGI (calling end user name)			
K1SSI (ICAM session ID)			
K1DEV (device ID of batch)		reserved	
K1FU (from user type)	K1RW2 (remote WS)	KRSID (WS rid/sid)	
K1FL6 (flags)		reserved	

CLOSE DATAGRAM

$	$	KQTYP (function code)	← KQTY2 (KCLOSE)
K2SSU (user session ID)			MYID
number of sessions		reserved	
reserved			
reserved			

ABORT DATAGRAM

$	$	KQTYP (function code)	← KQTY3 (KABRT)
K3SSU (user session ID)			MYID
K3RSC (reason code)		reserved	
K3DQ (DMI packets)			
reserved			

OPNACC DATAGRAM

$	$	KQTYP (function code)	← KQTY4 (KOPAC)
K4SSU (user session ID)			MYID
K4SSI (ICAM session ID)			
K4FU (from user type)			
reserved			
reserved			

OPNREJ DATAGRAM

$	$	KQTYP (function code)	← KQTY5 (KOPRJ)
K5SSU (user session ID)			MYID
K5RSC (reason code)		reserved	
reserved			
reserved			

CLSCNF DATAGRAM

$	$	KQTYP (function code)	← KQTY6 (KCLCF)
K6SSU (user session ID)			MYID
number of sessions		unused	
K6DQ (DMI packets returned)			
reserved			

NOTE:

Each control datagram label is prefixed by a 1- to 3-character label you specify in your CONTDTG DSECT call.

Figure 2-39. Control Datagram Label Format

Table 2-47. Control Datagram Parameter List Detailed Field Descriptions (Part 1 of 2)

Byte	Label*	Type and Length	Content	Word
0	K$$	H	Must be $$ to be a control datagram	1
2	KQTYP	H	Function code	
3	(KQFUN) KQTY1 (KOPEN) KQTY2 (KCLOS) KQTY3 (KABRT) KQTY4 (KOPAC) KQTY5 (KOPRJ) KQTY6 (KCLCF)	(XL1)	(Function code) Open request Close request Abort request Open accept Open reject Close confirm	
For Open Datagram				
4	K1CGI	F	From name	2
8	K1SSI	F	ICAM session ID	3
12	K1DEV	H	Device ID of batch	4
14	K1RE	H	Reserved	
16	K1FU KREMT KTERM KLCUP KPRCS KTELC KPDNS	X	From user type	5
17	K1RW2 KPRSC KSESC KSZSL KATA KFNKY KSYRQ KSTRV KDTRV	X	Remote workstation user	
18	KRSID	H	Rid/sid of workstation	
20	K1FL6	X	Flags	6
		X	Reserved	
21	K1NQS	H	Reserved	
For Close Datagram				
4	K2SSU	F	User's session ID	2
8	K2ASC	H	Active session count	3
10	K2R1	H	Reserved	
12	K2R2	2F	Reserved	4, 5

Table 2-47. Control Datagram Parameter List Detailed Field Descriptions (Part 2 of 2)

Byte	Label*	Type and Length	Content	Word
For Abort Datagram				
4	K3SSU	F	User's session ID	2
8	K3RSC	H	Reason code	3
10	K3R1	H	Reserved	
12	K3DQ	F	DMI packets	4
16	K3R2	F	Reserved	5
For OPNACC Datagram				
4	K4SSU	F	User's session ID	2
8	K4SSI	F	ICAM's session ID	3
12	K4FU	X XL3	User type Reserved	4
16	K4NQS	H H	Reserved Reserved	5
20	K4R1	F	Reserved	6
For OPNREJ Datagram				
4	K5SSU	F	User's session ID	2
8	K5RSC	H	Reason code	3
10	K5R1	H	Reserved	
12	K5R2	2F	Reserved	4, 5
For CLSCNF Datagram				
4	K6SSU	F	User's session ID	2
8	K6ASC	H	Active session count	3
10	K6R1	H	Reserved	
12	K6DQ	F	DMI packets returned	4
16	KQR2	F	Reserved	5
	KLSZ	EQU	Length of table	

* Labels are prefixed by default prefix I or by a 1- to 3-character prefix assigned by MF parameter on the CONTDTG DSECT macro call.

2.24.3.1. Datagram Types

The following describes the conditions under which session control datagrams are submitted to a user program. Datagrams are placed in the user's GAWAKE input buffer by ICAM and control is given to the user program at its GAWAKE entry address when it is in a communications CYIELD state.

- OPEN Datagram

  This datagram is passed to a user program when:

  - A terminal operator enters the $$SON command.
  - Another user program issues a SESCON macroinstruction with FUNCT=OPEN.

- CLOSE Datagram

  This data is passed to the user program when:

  - The closing of this session is requested by the calling user program with which the called user program is in session.

- ABORT Datagram

  This datagram is given to a user program when:

  - A GUST shutdown is initiated and the session is active.
  - The console operator marked the terminal down.
  - An error occurred and the terminal or line is down.
  - The terminal operator requested it via the $$SOFF type-in.
  - A user program in session with a second user program issues an NDETACH or FUNCT=ABORT; the second user program receives the ABORT datagram.

    When a session path between a terminal and another end user is aborted due to a $$SOFF message by the terminal operator, a line or terminal marked down in error, or the user program's issuing a SESCON macroinstruction with FUNCT=ABORT, the following occurs so that the terminal can establish a new session:

    1. All of the output messages destined for the terminal are removed and discarded.

2. All of the messages on the terminal input queue are discarded.

3. The terminal table is restored to its generative state.

- OPNACC Datagram

  This datagram is given to the user program when:

  - The user program issues a SESCON macroinstruction to another user program with FUNCT=OPEN and the requested user program accepts the open request.

  - The user program issues a SESCON macroinstruction to a process file or terminal and ICAM accepts the open request for the process file or terminal.

- OPNREJ Datagram

  This datagram is given to the user program when:

  - The user program issues a SESCON macroinstruction with FUNC=OPEN, and the requested user program does not accept the open request.

  - The required resources are not available for the session.

- CLSCNF Datagram

  This datagram is passed to the CUP when:

  - The calling user program previously requested the session with a called user program to be closed. This is the response returned by the called user program or on behalf of the called terminal or process file to acknowledge the close request.

2.24.3.2. Datagram Parameter Descriptions

The following are descriptions of each of the fields of the session control datagrams. (See Table 2-48.) The control datagrams differ from other user program generated datagrams in that the ICAM-supplied datagrams always begin with $$ followed by the datagram 2-character type code. Table 2-48 summarizes the parameter descriptions and lists the types of datagram in which each parameter appears.

- Calling End User Name

  Identifies the terminal or user program that initiated the open request.

- User Session Identifier (MYID)

  This parameter is defined to ICAM initially by the user program with the MYID parameter on the OPEN or OPNACC SESCON macroinstruction. It identifies to the user program the specific session in all of the control datagrams relating to the session except the OPEN datagram.

Table 2-48. Datagram Parameter Descriptions

Parameter	Datagram Type	Description
Calling end user name	OPEN	Identifies terminal or user program initiating open request.
ICAM session identifier	OPEN	Identifies session with parameter assigned by ICAM. Required by ICAM in all SESCON macroinstructions relating to session other than those using FUNCTOPEN.
User session identifier (MYID)	CLOSE	Identifies session with parameter assigned by user in SESCON macroinstruction using FUNCT=OPEN or OPNACC. Parameter supplied to user program in all related control datagrams.
Active session count	CLOSE	Indicates number of sessions that user program receiving datagram is currently involved in. Does not include the session being closed (CLOSE datagram) or that has been closed (CLSCNF datagram).

- ICAM Session Identifier

  ICAM-supplied 4-character identifier that the user program must, in turn, supply (in the associated parameter list) to ICAM in all SESCON macroinstructions regarding this session, except OPEN.

- Active Session Count

  A hexadecimal representation of the number of sessions that the user program receiving the datagram is still involved in. This does not include the session being closed (CLOSE datagram) or that has been closed (CLSCNF datagram).

2.24.3.3. Control Datagram and SESCON DSECTs

The calling sequence for obtaining a DSECT to cover a datagram is:

```
name    CONTDTG MF=(D,xxx)
```

where:

name
: Is an 8-character alphanumeric name that is used to reference the 2DSECT.

D
: Indicates a DSECT request.

xxx
: Is any 3-character alphanumeric prefix that makes labels generated in the DSECT unique.

Similarly, to obtain a DSECT for a user packet generated by a SESCON call, the following call may be used:

```
name    SESCON MF=(D,xxx)
```

2.24.3.4. Sample Dynamic Session User Program

Figure 2-40 is a working example of dynamic session user program. It is the same basic program used in 2.23.2.2 for a dedicated network modified to run in a global network environment.

```
USRDYN   START 0
         TN#DSECT DUST
         TN#DSECT GAWAKE
         TM#DSECT GETPUT
         BALR  10,0
         USING *,10
*                 **********************************
*                 ****** COVER INPUT DTFCP ******
*                 **********************************
         USING TM#PRCS,2
         LA    2,DUMY
*                 **********************************
*                 ****** COVER OUTPUT DTFCP *****
*                 **********************************
         USING TM#DEST,3
         LA    3,OTPT
*                 **********************************
*                 ****** REQUEST A NETWORK *******
*                 **********************************
BEGIN    NATTACH NET1,ERRET=NETERR,APPS=CUP1
         GAWAKE TYPE=INPUT,ENTRY=DGENTRY,BUFADR=DATAGRAM,                X
               BUFLTH=40,APPS=CUP1
         SLL   0,24                    SHIFT OUT ALL BUT GAWAKE ERRS
         ST    0,CHECK                 STORE IN CHECK AREA
         CLC   CHECK,CLEAR             CHECK FOR ERRORS
         BNE   DISPGREJ                ANY ERRORS GO DISPLAY
         CYIELD                        NO ERRORS WAIT FOR DATAGRAM
*                 **********************************
*                  DATAGRAM ENTRY TO INTERROGATE
*                     DYNAMIC SESSION REQUESTS
*                 **********************************
DGENTRY  LA    1,DATAGRAM               MAP DATAGRAM BUFFER
         USING DSDTGRM,1                COVER REGISTER FOR DSECT
         CLI   DATKQFUN,DATKOPEN         IS IT AN OPEN?
         BE    OPNDAT                   YES
         CLI   DATKQFUN,DATKCLOS          NO-IS IT A CLOSE?
         BE    CLCFDAT                  YES
         CLI   DATKQFUN,DATKABRT          NO-IS IT AN ABORT?
         BE    ABRTDAT                  YES
         CLI   DATKQFUN,DATKOPAC          NO-IS IT AN OPEN ACCEPT?
         BE    ACCPTDAT                 YES
         CLI   DATKQFUN,DATKOPRJ         NO-IS IT AN OPEN REJECT?
         BE    REJDAT                   YES
         CLI   DATKQFUN,DATKCLCF          NO-IS IT A CLOSE CONFIRM?
         BE    CLOSDAT                  YES
         B     INVALID                  NO-IT'S NOT ANY VALID
*                                       FUNCTION GO DISPLAY
*                 **********************************
*                   DATAGRAM PROCESSING ROUTINES
*                 **********************************
OPNDAT   EQU   *                        OPEN DATAGRAM
         MVC   EUTAG,DATK1CGI           SAVE END USER NAME
         MVC   ICSESSID,DATK1SSI        SAVE ICAM SESSION ID
         B     OPNACC                   RESPOND WITH ACCEPT
*
CLOSDAT  EQU   *                        CLOSE DATAGRAM
         MVC   MYSESSID,DATK2SSU        SAVE USER SESSION ID
         B     CLCFSES                  RESPOND WITH CLOSE CONFIRM
*
ABRTDAT  EQU   *                        ABORT DATAGRAM
         MVC   MYSESSID,DATK3SSU        SAVE USER SESSION ID
         B     ABORT                    GO TO END OF SESSION
```

Figure 2-40. Dynamic Session User Program (Part 1 of 6)

```
*
ACCPTDAT EQU   *                          ACCEPT DATAGRAM
         MVC   ICSESSID,DATK4SSI          SAVE ICAM SESSION ID
         MVC   MYSESSID,DATK4SSU          SAVE USER SESSION ID
         B     GOPUT                      GO TO GET/PUT ROUTINE
*
REJDAT   EQU   *                          OPEN REJECT DATAGRAM
         MVC   MYSESSID,DATK5SSU          SAVE USER SESSION ID
         B     DISPREJ                    MY OPEN REJECTED-GO DISPLAY
*
CLCFDAT  EQU   *                          CLOSE CONFIRM DATAGRAM
         MVC   MYSESSID,DATK6SSU          SAVE USER SESSION ID
         B     ENDJOB                     GO TO NORMAL CLOSE
*          **********************************
*               SESCON BUILD ROUTINES
*          **********************************
OPNSES   LA    1,SES1                       PROGRAM WANTS TO INIT SES
         USING DSESCON,1                  MAP SESCON
         MVI   SESFUNC,SESTOPEN           SET TO OPEN SESSION
         MVI   SESTQLTH,SESS1LT           SET PROPER LENGTH
         MVC   SESS1CI,EUTAG              INSERT END USER NAME
         MVC   SESS1SSU,MYSESSID          INSERT MY USER ID
         MVC   SESS11PQ,ZEROES            CLEAR INPUT QUEUE FIELD
         SESCON MF=(E,(1))                 ISSUE SESCON
         SLL   0,24                       SHIFT TO KEEP ONLY SES ERRS
         ST    0,CHECK                    STORE IN CHECK AREA
         CLC   CHECK,CLEAR                CHECK FOR ERRORS
         BNE   DISPOPN                    ANY ERROR GO DISPLAY
         CYIELD                           NO ERRORS AWAIT REPLY
*
CLSSES   LA    1,SES1                       PROGRAM WANTS TO CLOSE SES
         USING DSESCON,1                  MAP SESCON
         MVI   SESFUNC,SESTCLOS           SET TO CLOSE SESSION
         MVI   SESTQLTH,SESS2LT           SET PROPER LENGTH
         MVC   SESS2SSI,ICSESSID          INSERT ICAM SESSION ID
         SESCON MF=(E,(1))                 ISSUE SESCON
         SLL   0,24                       SHIFT TO KEEP ONLY SES ERRS
         ST    0,CHECK                    STORE IN CHECK AREA
         CLC   CHECK,CLEAR                CHECK FOR ERRORS
         BNE   DISPCLS                    ANY ERRORS GO DISPLAY
         CYIELD                           NO ERRORS AWAIT CONFIRM
*
ABRTSES  LA    1,SES1                       PROGRAM WANTS TO ABORT SES
         USING DSESCON,1                  MAP SESCON
         MVI   SESFUNC,SESTABRT           SET TO ABORT
         MVI   SESTQLTH,SESS3LT           SET PROPER LENGTH
         MVC   SESS3SSI,ICSESSID          INSERT ICAM ID
         SESCON MF=(E,(1))                 ISSUE THE FUNCTION
         SLL   0,16                       SHIFT TO KEEP ONLY SES ERRS
         ST    0,CHECK                    STORE IN CHECK AREA
         CLC   CHECK,CLEAR                CHECK FOR ERRORS
         BNE   DISPABT                    ANY ERRORS GO DISPLAY
         B     DONEJOB                    GO TO SNAP AND END
*
OPNACC   LA    1,SES1                       ACCEPT END USER REQ TO OPEN
         USING DSESCON,1                  MAP DSECT
         MVI   SESFUNC,SESTOPAC           SET TO OPEN ACCEPT
         MVI   SESTQLTH,SESS4LT           SET PROPER LENGTH
         MVC   SESS4SSI,ICSESSID          INSERT ICAM ID
         MVC   SESS4SSU,MYSESSID              INSERT MY USER ID
         SESCON MF=(E,(1))                 ISSUE SESCON
```

Figure 2-40. Dynamic Session User Program (Part 2 of 6)

```
          SLL   0,24                      SHIFT TO KEEP ONLY SES ERRS
          ST    0,CHECK                   STORE IN CHECK AREA
          CLC   CHECK,CLEAR               CHECK FOR ERRORS
          BNE   DISPACC                   NO-GO DISPLAY
          B     GOPUT                     GO TO GET/PUT ROUTINE
*
OPNREJ    LA    1,SES1                    OPEN REJECT REQUEST
          USING DSESCON,1                 MAP DSECT
          MVI   SESFUNC,SESTOPRJ          SET TO OPEN REJECT
          MVI   SESTQLTH,SESS5LT          SET PROPER LENGTH
          MVC   SESS5SSI,ICSESSID          INSERT ICAM SESSION ID
          SESCON MF=(E,(1))                    ISSUE SESCON
          SLL   0,24                      SHIFT TO KEEP ONLY SES ERRS
          ST    0,CHECK                   STORE IN CHECK AREA
          CLC   CHECK,CLEAR               CHECK FOR ERRORS
          BNE   DISPREJ                   NO-GO DISPLAY
          CYIELD                          WAIT FOR NEW DATAGRAM
*
CLCFSES   LA    1,SES1                    CONFIRM CLOSE REQUEST
          USING DSESCON,1                 MAP DSECT
          MVI   SESFUNC,SESTCLCF          SET TO CONFIRM
          MVI   SESTQLTH,SESS6LT          SET TO PROPER LENGTH
          MVC   SESS6SSI,ICSESSID         INSERT ICAM SESSION ID
          SESCON MF=(E,(1))                   ISSUE SESCON
          SLL   0,24                      SHIFT TO KEEP ONLY SES ERRS
          ST    0,CHECK                   STORE IN CHECK AREA
          CLC   CHECK,CLEAR               CHECK FOR ERRORS
          BNE   DISPCLCF                  ANY ERRORS GO DISPLAY
          B     CLOSE                     GO TO NORMAL END
*         **********************************
*              SET UP FIRST PUTCP
*         **********************************
GOPUT     EQU   *                         START OF GET/PUT ROUTINE
LOOP      OI    TM#DSEG,TM#DHDR+TM#DTND   SET FOR COMPLETE MESSAGE
          XC    TM#DERR,TM#DERR           CLEAR ERROR BYTE AREA
          MVC   TM#DENA,EUTAG             INSERT DESTINATION NAME
PUTIT1    PUTCP OTPT,MSGOT1               ISSUE PUT
*         **********************************
*              SET UP SECOND PUTCP
*         **********************************
          OI    TM#DSEG,TM#DHDR+TM#DTND
          XC    TM#DERR,TM#DERR
          MVC   TM#DENA,EUTAG             INSERT DESTINATION NAME
PUTIT2    PUTCP OTPT,MSGOT2
*         ******************************
*         SET UP DEFERRED GETCP
*         ******************************
          XC    TM#PERR,TM#PERR
          MVI   TM#PIND,TM#PIRL
          MVC   TM#PCMPA,=A(BINGO)
          MVC   TM#PNAM,EUTAG             INSERT INPUT DTF NAME TO USE
          MVC   MSGIN,TXTCHARS
GETIT     GETCP DUMY,MSGIN
*                 **********************************
*                 ** YIELD FOR ANY ACTIVITY ***
*                 **********************************
          CYIELD
*                 **********************************
*                 **** WHAT KIND OF MESSAGE? ****
*                 ****     DONE OR A NAME?    ****
*                 **********************************
```

Figure 2-40. Dynamic Session User Program (Part 3 of 6)

```
BINGO    SNAP  MSGOT1,CHECK+3
         CLC   TXTIN(4),=C'DONE'
         BE    ENDJOB
         CLC   TXTIN(4),=X'84969585'
         BE    ENDJOB
*        **********************************
*                ECHO INPUT TO OUTPUT
*        **********************************
         MVC   TXTOUT,TXTIN
         MVC   TXTIN,C' '
         MVC   TXTIN+1(L'TXTIN-1),TXTIN
*        **********************************
*        SET UP PUTCP TO ECHO MESSAGE
*        **********************************
         OI    TM#DSEG,TM#DHDR++TM#DTND
         XC    TM#DERR,TM#DERR
         MVC   TM#DENA,EUTAG
PUTIT3   PUTCP OTPT,MSGOUT
         B     LOOP
*        **********************************
*           RELEASE THE GLOBAL NETWORK
*        **********************************
ENDJOB   NDETACH NET1
         B     DONE
*        **********************************
*          BUILD ERROR DISPLAY MESSAGES
*        **********************************
NETERR   LA    1,MSG1
         OPR   MSG1,27
         B     DONEJOB                    SNAP
NOGET    LA    1,MSG2
         OPR   MSG2,12
         B     DONEJOB                    SNAP
NOPUT    LA    1,MSG3
         OPR   MSG3,12
         B     DONEJOB                    SNAP
NOBUFF   LA    1,MSG4
         OPR   MSG4,19
         B     DONEJOB                    SNAP
DISPOPN  EQU   *
         LA    1,MSG5
         OPR   MSG5,18
         B     DONEJOB                    SNAP
DISPACC  EQU   *
         LA    1,MSG6
         OPR   MSG6,20
         B     DONEJOB                    SNAP
DISPREJ  EQU   *
         LA    1,MSG7
         OPR   MSG7,20
         B     DONEJOB                    SNAP
DISPCLS  EQU   *
         LA    1,MSG8
         OPR   MSG8,19
         B     DONEJOB                    SNAP
DISPABT  EQU   *
         LA    1,MSGA
         OPR   MSG9,21
         B     DONEJOB                    SNAP
```

Figure 2-40. Dynamic Session User Program (Part 4 of 6)

```
DISPCLCF EQU   *
         LA    1,MSGA
         OPR   MSGA,21
         B     DONEJOB                    SNAP
INVALID  EQU   *
         LA    1,MSGB
         OPR   MSGB,25
         B     DONEJOB                    SNAP
ABORT    EQU   *
         LA    1,MSGC
         OPR   MSGC,36
         B     ENDJOB                     DONE
CLOSE    EQU   *
         LA    1,MSGD
         OPR   MSGD,36
         B     ENDJOB                     DONE
DISPGREJ EQU   *
         LA    1,MSGE
         OPR   MSGE,20
         B     DONEJOB                    SNAP
DONEJOB  SNAP  BEGIN,CHECK+3
DONE     EOJ
*                ********************************
*                ****     CONSTANT AREA      ****
*                ********************************
DUMY     DTFCP TYPE=GT,ERRET=NOGET
OTPT     DTFCP TYPE=PT,ERRET=NOPUT,DEST=(T,DTAG),NOBAV=NOBUFF
SES1     SESCON FUNCT=OPEN,MYID=AAD1,TONAME=FILL,MF=L
DSESCON  SESCON MF=(D,SES)
DSDTGRM  CONTDTG MF=(D,DAT)
         DS    0F
MSG1     DC    CL26'ERROR ISSUING NATTACH MACRO'
MSG2     DC    CL12'ERROR ON GET'
MSG3     DC    CL12'ERROR ON PUT'
MSG4     DC    CL19'NO BUFFER AVAILABLE'
MSG5     DC    CL18'OPEN NOT VALIDATED'
MSG6     DC    CL20'ACCEPT NOT VALIDATED'
MSG7     DC    CL20'REJECT NOT VALIDATED'
MSG8     DC    CL19'CLOSE NOT VALIDATED'
MSG9     DC    CL19'ABORT NOT VALIDATED'
MSGA     DC    CL21'CONFIRM NOT VALIDATED'
MSGB     DC    CL25'INVALID FUNCTION RECEIVED'
MSGC     DC    CL36'ABORT RECEIVED - SIGN OFF NOW PLEASE'
MSGD     DC    CL36'CLOSE RECEIVED - SIGN OFF NOW PLEASE'
MSGE     DC    CL20'GAWAKE NOT VALIDATED'
         DS    0H
MSGOT1   DC    X'0019'
         DC    X'10030000'
         DC    C'WELCOME TO OS/3 ICAM'
         DC    X'0D'
         DS    0H
MSGOT2   DC    X'001A'
         DC    C'TYPE IN YOUR NAME PLEASE'
         DC    X'0D1E'
         DS    0H
MSGIN    DC    X'0000'
         DC    CL7' '
TXTIN    DC    CL256' '
INETX    DC    X'FFFF'
         DS    0H
```

Figure 2-40. Dynamic Session User Program (Part 5 of 6)

```
MSGOUT    DC     X'010C'
          DC     X'0D'
          DC     C'THANK YOU '
TXTOUT    DC     CL256' '
OUTETX    DC     X'0D'
TXTCHARS  DC     X'0105'
ICSESSID  DC     CL4' '                    ICAM SUPPLIED SESSION ID
MYSESSID  DC     C'AA01'                   MY SESSION ID
ZEROES    DC     XL4'0'
CLEAR     DC     XL4'0'
EUTAG     DC     CL4' '                    NAME OF END USER IN SESSION
          DS     0F
DATAGRAM  DC     40' '                     DATAGRAM BUFFER AREA
CHECK     DC     XL4'0'                    ERROR BYTE STORAGE AREA
*                                          FOR GAWAKE AND SESCON
*                                          VALIDATION
          END
```

Figure 2-40. Dynamic Session User Program (Part 6 of 6)

2.25. ICAM ERROR MESSAGES

When ICAM cancels your program, the system console operator is notified by a JC03 message. This message also contains an error code describing why your program was canceled. See the system messages programmer/operator reference, UP-8076 (current version) for a description of the JC03 message and the ICAM error codes.

3. Remote Terminal Characteristics

3.1. GENERAL

This section describes the characteristics and handling of the various remote terminals that are supported by the ICAM communications environment.

Remote terminals are supported under ICAM in either an interactive or batch environment.

Table 3-1 lists those terminals that are supported in each environment.

Table 3-1. Interactive/Batch Mode Terminal Support (Part 1 of 2)

Terminal and Common Name	Mode of Operation	
	Interactive	Batch
UNISCOPE 100 Display Terminal (UNISCOPE 100)	X	
UNISCOPE 200 Display Terminal (UNISCOPE 200)	X	
SPERRY UNIVAC Universal Distributed System 2000 (UDS 2000) (Operating in UNISCOPE mode)	X	
SPERRY UNIVAC Universal Terminal System 400 (UTS 400) (Operating in UNISCOPE 100 mode)	X	
SPERRY UNIVAC Universal Terminal System 400 (UTS 400) (Operating in native mode)	X	
SPERRY UNIVAC Universal Terminal System 400 Text Editor (TE)	X	
SPERRY UNIVAC Universal Terminal System 4000 (UTS 4000): UTS 20 and UTS 40 Terminals	X	
TELETYPE Corporation teletypewriter (TTY)	X	
IBM 3270 Terminal System	X	
SPERRY UNIVAC Data Communications Terminals DCT 500, DCT 475, DCT 524	X	
SPERRY UNIVAC Universal Terminal System 10 (UTS 10) (Single station operating in TTY mode)	X	

Table 3—1. Interactive/Batch Mode Terminal Support (Part 2 of 2)

Terminal and Common Name	Mode of Operation	
	Interactive	Batch
SPERRY UNIVAC Data Communications Terminal 1000 (DCT 1000)	X	X
SPERRY UNIVAC Data Communications Terminal 2000 (DCT 2000)		X
SPERRY UNIVAC 1004 Card Processor System (1004 card processor)		X
SPERRY UNIVAC 9200/9300 Series (9200/9300 subsystem)		X
IBM 2780 Data Communications Terminal (IBM 2780)		X
IBM 3741 Data Communications Terminal		X
SPERRY UNIVAC Universal Distributed System 2000 (UDS 2000) (Emulating an IBM 2780/3741)		X
Binary synchronous procedures (BSC) EBCDIC transparent and ASCII nontransparent		X

3.2. COMMON AREAS OF INTERACTIVE AND BATCH MODE ENVIRONMENTS

The user program interface to ICAM is defined:

- to create output to be sent to remote terminals; and
- to process input received from remote terminals.

The interface definition can be complicated by any of the following:

- Output text may be sent to the primary output device of a remote terminal or to any of several possible auxiliary output devices.
- Input text may be received from the primary input device of a remote terminal or from any of several possible auxiliary input devices.
- Special output commands may be sent to the remote terminal.
- Special input commands may be received from the remote terminal hardware or the remote terminal operator.

User programs may communicate via an RDH with remote terminals. In executing the required input and output functions, an RDH performs manipulation of the data as follows:

- Removes and builds an envelope
- Performs translation between EBCDIC and the character code of the remote terminal
- Converts appropriate line and form control information
- Handles special input or output commands

An envelope consists of appropriate control characters required by the remote terminal, i.e., SOH, STX, ETX, etc. The RDH adds the envelope on output and strips it off on input. The user handles only the input or output text.

The line and forms control information is handled via DICE.

To perform the required input and output functions, two fields have been defined for use by user programs:

1. Auxiliary device index field

   Controls the destination of output or informs the user of the origination of input.

2. Auxiliary device/special function field

   Informs the user of special input commands from the remote terminal or sends special output commands to the remote terminal.

These two fields have different naming conventions, depending on the type of user interface program used and on output versus input.

3.2.1. Common Output Specifications

The location of the auxiliary device index field and auxiliary device/special function field for output is illustrated in Table 3-2.

Table 3-2. Interface Packet Output Auxiliary Device Index and Auxiliary Device/Special Function Fields

Interface	Packet Containing Specified Field	Auxiliary Device Index Field	Auxiliary Device/ Special Function Field
STDMCP	Destination queue DTFCP	TM#DDVC	TM#DSPEC

3.2.2. Common Input Specifications

The location of the auxiliary device index field and auxiliary device/special function field for input is illustrated in Table 3-3.

Table 3-3. Interface Packet Input Auxiliary Device Index and Auxiliary Device/Special Function Fields

Interface	Packet Containing Specified Field	Auxiliary Device Index Field	Auxiliary Device/ Special Function Field
STDMCP	Process file DTFCP	TM#PDVC	TM#PSPEC

3.2.3. CCA Generation

Certain characteristics of remote terminals are flagged at network generation by the TERM macroinstruction. These flags may affect the processing that is described in 3.3 and 3.4.

3.2.4. Error Handling

A set of error conditions can be reported to the user program and/or sent to the system console by ICAM. This section discusses only those errors handled by ICAM with which the RDH is involved. Section 2 of this manual describes:

- the error codes that ICAM may present to the user;
- the presentation format;
- the conditions required for the error codes to be presented; and
- disposition of messages on ICAM queues when the error occurs.

The RDHs are involved with the following types of error conditions:

- The RDH receives a primary status other than successful. The unsuccessful status situations may be:

  1. The primary status indicates an unrecoverable hardware error (TN#PHDWR). No attempts at error recovery are made by ICAM. The line is immediately marked down.

  2. The primary status indicates an error for which attempts at error recovery can be made. These errors include:

     - No response by the terminal (software time-out)
     - Input message hit on the line (input parity)

- The RDH receives a primary status from CPI of successful completion (TN#PEND), but the RDH detects an error condition. The situations may be:

  1. Bad site-ID/terminal address from the terminal; does not match the network-defined site-ID/terminal address. Attempts at error recovery are made for the UNISCOPE terminal but not for the other terminals.

  2. The contents of the input message header are invalid and cannot be decoded by the RDH. Attempts at error recovery are made.

  3. Negative acknowledge on output. The output may have been hit on the line, creating a parity error. The terminal does not accept the output and informs the processor with a negative acknowledge. Attempts at error recovery are made.

Error handling for errors peculiar to a particular remote terminal is discussed in the sections describing each remote terminal.

3.2.4.1. Terminal Up/Down Conditions

When ICAM marks a terminal down, the following information is included in a message sent to the system console:

TERMINAL NAME - (REASON) . TERM DOWN.

When ICAM marks a terminal up, the following information is included in a message sent to the system console:

TERMINAL NAME - DOWN TERM MARKED UP.

ICAM marks a terminal down under the following conditions:

- The error was other than an unrecoverable hardware error.
- Attempts at error recovery were made, and they were unsuccessful.

When a terminal is marked down, input solicitation (polling) by ICAM continues automatically. However, ICAM stops sending output to the down terminal. When ICAM receives input from the down terminal, that terminal is marked up. The input is scheduled to the user.

3.2.4.2. Line Error Notification

ICAM notifies you of an abnormal line condition; you are scheduled at the ERRET address specified in the NETREQ macroinstruction. Register 1 contains the line name that contains the error.

Table 3-4 contains the possible error codes for a STDMCP user.

Table 3-4. Line Level Notification Error Codes

Type	Reason for Line Level Notification	STDMCP (Byte 1 of Register 0)
1	Line disconnected due to an unrecoverable line error.	TM#DNLNO
2	Line down due to final terminal on the line being marked down.	TM#DNDNA
3	Bad site-id for 1004. Invalid sid for UTS 400/UTS 4000/UNISCOPE. Invalid did for DCT1000.	TM#DNSIT

Table 3-5 describes the actions taken for each type of line-down notification.

Table 3-5. Actions Due to Line-Down Notification

Type	System Console Message Provided by ICAM	Action by ICAM	Suggested User Action
1	MC#16 MC#61	- Polling is stopped. - Output is stopped.	LNEREQ to clear message and start polling
2	MC#50 to MC#53 MC#56 to MC#60	- Polling continues. - Output is stopped. - Input causes all terminals on the line to be marked up.	
3	MC#14	- Polling continues. - Output continues.	LNEREL to protect unauthorized access to user files

Consult the system messages programmer/operator reference, UP-8076 (current version), for a description of the terminal up/down and line up/down messages that can be sent to the system console, and for a description of appropriate console commands available to the SPERRY UNIVAC 90/30 System operator.

3.3. INTERACTIVE MODE TERMINALS

This subsection discusses the remote terminal characteristics for those remote terminals that operate in an interactive environment. Characteristics common to each interactive terminal are illustrated, followed by a discussion of each supported interactive terminal.

3.3.1. Characteristics of Interactive Terminals

Interactive terminals have the following characteristics:

- The supported interactive terminals are characterized by keyboard input.
- Output is normally processed by a character-oriented printer or a CRT screen.
- Some I/O devices are designated primary input or output devices. In addition to the primary I/O devices, a selection of auxiliary I/O devices may be supported, as indicated in Table 3-6.

Table 3-6. Auxiliary Devices Supported for Specific Terminals

Interactive Terminal	Primary Devices	Auxiliary Devices
UNISCOPE	Keyboard/screen	Tape cassette system (TCS), communications output printer (COP), 800 terminal printer (TP), 0786 printer
DCT 1000	Keyboard/printer	Card reader, card punch Paper tape reader/punch
DCT 500	Keyboard/printer	Paper tape reader/punch
DCT 524	Keyboard/printer	Tape cassette system (TCS) (read and write functions only)
DCT 475	Keyboard/printer	None
TTY (33,35,37)	Keyboard/printer	Paper tape reader/punch
UTS 400	Keyboard/screen	Diskette, TCS, COP, 800 TP, 0786 printer
UDS 2000	Integral diskette (also keyboard/screen)	0786 printer, diskette subsystem
IBM 3270	3277 display 3284 printer 3286 printer	None
UTS 4000	Keyboard/screen	Diskette, 0797/0798 printer, 0791 correspondence quality printer, magnetic stripe reader

3.3.1.1. Interactive Terminal Output

Table 3-7 specifies the types of output that the user program can submit to ICAM. It specifies that the user must supply to ICAM:

- text when required;
- contents of the special function field; and
- contents of the auxiliary device index field.

It also lists special RDH considerations.

NOTE:

When a UTS 400/UTS 4000 terminal is configured, the acronym TCS refers to the tape cassette subsystem (TCS) or the diskette; the acronyms TP or COP refer to a 800 terminal printer.

Table 3-7. Output Description for Interactive Terminals (Part 1 of 6)

Terminal	Desired Result (1) (Output Type)	Special RDH Considerations	Must User Supply Text? (3)	Contents of Auxiliary Device Index Field (TM#DDVC)	Contents of Special Function Field (5) (TM#DSPEC)
UNISCOPE 100/200 UTS 400 /UTS 4000	Normal output to the screen	DICE is applied for the screen.	Yes	0	0
	Output to the screen. The data is a user-supplied form that has significant blank segments that include the last column of the screen. The terminal operator fills in the form and presses the transmit key.	DICE is applied for the screen. All spaces are converted to DC3 characters in the user-supplied text. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	0	TM#DPRE
	Output to an auxiliary device that has the same format as the screen. This is print mode.	DICE is applied for the screen. The print mode character (DC2) is inserted at the end of the text. (2)	Yes	(4)	TM#DADWR
	Output to an auxiliary device that has a format independent of the screen. This is print transparent mode.	DICE is applied for the COP. The print transparent sequence (ESC DC2) is inserted at the end of the text. (2)	Yes	(4)	TM#DADWR ++ TM#DTAT
	Output to an auxiliary device that has a format independent of the screen. The user-supplied data has significant blank segments that include the last column of the screen.	DICE is applied for the COP. The print transparent sequence (ESC DC2) is inserted at the end of the text. The RDH changes all spaces to DC3 characters. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	(4)	TM#DADWR ++ TM#DTAT ++ TM#DPRE
	TCS read that transfers the next block of data from the TCS to the screen. (6)	The READ command is built by the RDH.	No	(4)	TM#DADRD
	TCS read, so that the block of data from the TCS will appear to your program as identical to that which your program previously supplied to ICAM to be written on the TCS. (6)				TM#DADRD ++ TM#DTAT

Table 3-7. Output Description for Interactive Terminals (Part 2 of 6)

Terminal	Desired Result (1) (Output Type)	Special RDH Considerations	Must User Supply Text? (3)	Contents of Auxiliary Device Index Field (TM#DDVC)	Contents of Special Function Field (5) (TM#DSPEC)
UNISCOPE 100/200 UTS 400 /UTS 4000 (cont)	TCS search. The mode A, B, or C search results in the transfer of the found block of data from the TCS to the screen. The @ mode search is for tape positioning only. (6) (9)	The user-supplied text provides the search mode and address/ identifier which is used by the RDH to build the SEARCH command.	Yes (7)	(4)	TM#DADSR
	TCS mode A, B, or C search so that the block of data from the TCS will appear to your program as identical to that which your program previously supplied to ICAM to be written on the TCS. (6) (9)				TM#DADSR ++ TM#DTAT
	TCS report address. The previous TCS operation was a TCS write. The report address results in the transfer to the screen of the address of the block of data that was just written. If the previous TCS operation was other than write, the transferred address is the current TCS (6) position as shown by the (9) address indicator on the TCS.	The REPORT ADDRESS command is built by the RDH.	No	(4)	TM#DADRA
	TCS backward one block (9)	The backward-one-block command is built by the RDH.	No	(4)	TM#DADBS
	Light the computer message waiting indicator and sound the audible alarm	The COMPUTER MESSAGE WAITING command is built and sent by the RDH.	No	0	TM#DFSMW or TM#DFMWS (1)
UTS 400 /UTS 4000 (not applicable to UNISCOPE mode)	Disconnect terminal	RDH builds the disconnect sequence (DLE EOT STX ETX).	Yes (DLE EOT)	0	0
	Initiate confidence test at terminal	Ignore poll response time-out while confidence test is running.	Yes (ESC Q) (10)	0	0
	Call error log from terminal	None	Yes (ESC P) (10)	0	0
	Clear error log in terminal	None	Yes (ESC R) (10)	0	0

Table 3-7. Output Description for Interactive Terminals (Part 3 of 6)

Terminal	Desired Result (1) (Output Type)	Special RDH Considerations	Must User Supply Text? (3)	Contents of Auxiliary Device Index Field (TM#DDVC)	Contents of Special Function Field (5) (TM#DSPEC)
UTS 400/ UTS 4000 (cont)	Load program for execution (11)	RDH will not translate.	Yes (9)(10)	0	0
	Load program on diskette (11)	RDH will not translate.	Yes	(4)	TM#DADWR ++ TM#DTAT
	Cause terminal to send cursor address	None	Yes (ESC T)	0	0
	Request dump from terminal (11)	RDH will not translate.	Yes (10)	0	0
	Output to an auxiliary device of all unprotected characters in the field to be printed (SOE to cursor). Protected characters are changed to spaces.	DICE is applied for the COP. The print form (ESC H) is inserted at the end of the text.	Yes	(4)	TM#DADPF
	Output to an auxiliary device of all unprotected characters in the field to be printed (SOE to cursor). Protected characters are changed to spaces. The user-supplied data has significant blank segments that include the last column of the screen.	DICE is applied for the COP. The print form (ESC H) sequence is inserted at the end of the text. The RDH changes all spaces to DC3 characters. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	(4)	TM#DADPF ++ TM#DPRE
	Output to an auxiliary device of all characters between the SOE character and the cursor including FCC sequences.	DICE is applied for the COP. The transfer all (ESC G) sequence is inserted at the end of the text.	Yes	(4)	TM#DADXA
	Output to an auxiliary device of all characters between the SOE character and the cursor including FCC sequences. The user-supplied data has significant blank segments that include the last column of the screen.	DICE is applied for the COP. The transfer all sequence (ESC G) is inserted at the end of the text. The RDH changes all spaces to DC3 characters. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	(4)	TN#DADPF ++ TM#DPRE
	Output to an auxiliary device of only the variable (not protected) characters between the SOE character and the cursor including FCC sequences.	DICE is applied for the COP. The transfer-variable sequence (ESC F) is inserted at the end of the text.	Yes	(4)	TM#DADXV

Table 3-7. Output Description for Interactive Terminals (Part 4 of 6)

Terminal	Desired Result (1) (Output Type)	Special RDH Considerations	Must User Supply Text? (3)	Contents of Auxiliary Device Index Field (TM#DDVC)	Contents of Special Function Field (5) (TM#DSPEC)
UTS 400/ UTS 4000 (cont)	Output to an auxiliary device of only the variable characters between the SOE character and the cursor including FCC sequences. The user-supplied data has significant blank segments that include the last column of the screen.	DICE is applied for the COP. The transfer variable sequence (ESC F) is inserted at the end of the text. The RDH changes all spaces to DC3 characters. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	(4)	TM#DADXV ++ TN#DPRE
	Output to an auxiliary device of only the changed fields (characters) between the SOE character and cursor including FCC sequences.	DICE is applied for the COP. The transfer changed sequence (ESC E) is inserted at the end of the text.	Yes	(4)	TM#DADXC
	Output to an auxiliary device of only the changed fields (characters) between the SOE character and the cursor including FCC sequences. The user-supplied data has significant blank segments that include the last column of the screen.	DICE is applied for the COP. The transfer changed sequence (ESC E) is inserted at the end of the text. The RDH changes all spaces to DC3 characters. The clear function is done with DC3 characters on the current position control and clear DICE. (2)	Yes	(4)	TM#DADXC ++ TM#DPRE
UDS 2000	Normal output to the screen	(13)	Yes	0	0
	Output to an integral diskette or an auxiliary device. This is print mode.	Print mode character (DC2) is inserted. (2) (13)	Yes	(4) (12)	TN#DADWR
	TCS read that transfers multiple records from integral diskette or an auxiliary device to UDS 2000 main storage.	The READ command is built by the RDH.	No	(4) (12)	TM#DADRD
DCT 1000	Light computer message waiting indicator and sound audible alarm	Computer message waiting command is built and sent by the RDH.	No	0	TM#DFSMW (1)
DCT1000, DCT 500, DCT 524, DCT 475, TTY, or UTS 10	Normal output to the primary device (printer)	DICE is applied to the primary device (printer).	Yes	0	0

Table 3-7. Output Description for Interactive Terminals (Part 5 of 6)

Terminal	Desired Result (1) (Output Type)	Special RDH Considerations	Must User Supply Text? (3)	Contents of Auxiliary Device Index Field (TM#DDVC)	Contents of Special Function Field (5) (TM#DSPEC)
DCT 1000, DCT 500, or TTY	Output to the paper tape punch	DICE is applied to the primary device even though the output is to an auxiliary device. The RDH supplies an EM character at the end of text to the DCT 1000, if the text length is less than 160 characters.	Yes	(4)	TM#DADWR
DCT 524	Output text to the TCS write head				
DCT 1000	Output to the card punch				
DCT 500, DCT 524, DCT 475, TTY or UTS 10 (8)	Print/CMW or the MSGWAIT keyword parameter specified (four characters)	The 4-character output is built and sent by the RDH.	No	0	TM#DFSMW
DCT 500 or TTY	Paper tape read (6)	The RDH builds the command to cause reader to read preloaded tape. Reader switches must be on in order to respond.	No	(4)	TM#DADRD
DCT 524	TCS read (6)				
IBM 3270	Normal output to display printer	DICE is supplied.	Yes	0	0
	Print/CMW. Sound alarm or light the computer message waiting indicator	The computer message waiting command is built and sent by the RDH.	No	0	TM#DFSMW

NOTES:

(1) See 3.3.1.1.1 for a discussion of computer message waiting logic.

(2) See 3.3.1.1.2 for a discussion of UNISCOPE transmit and auxiliary device transfer functions.

(3) If the user program is required to supply text with the output and does not, an error indication will be given to the user.

(4) The auxiliary device index field must contain the auxiliary device index. Refer to the TERM macro AUXn parameter of the CCA network definition where n is the index number.

(5) See 3.3.1.1.3 for the EQU statements used to set the special function byte.

(6) See 3.3.1.1.4 for a discussion of soliciting auxiliary device input via output commands.

(7) See 3.3.1.1.5 for a listing of TCS search mode messages.

(8) Refer to the TERM macro MSGWAIT parameter of the CCA network definition.

Table 3-7. Output Description for Interactive Terminals (Part 6 of 6)

⑨ The search backward one block and report address commands to a diskette attached to a UTS 400 or UTS 4000 do not function correctly if the location of the argument or search mask representation on the screen coincide with a field control character (FCC) sequence. For example, if you issue one of these commands to a diskette and the command is displayed on the UTS 400/UTS 4000 screen where an FCC sequence is established, the command will not work correctly. Therefore, if you issue one of these commands to a diskette, you must clear any field control characters from the area of the UTS 400 screen where the commands may be directed. This consideration does not apply to the tape cassette system (TCS).

⑩ Must address the master or primary terminal

⑪ Not applicable to the UTS 400 Text Editor (TE) or the UTS 20 single station

⑫ UDS 2000 operator performs device selection

⑬ If your program supplies DICE sequences in text, the RDH will convert them to the required function. However, the UDS 2000 ignores these sequences.

3.3.1.1.1. Computer Message Waiting Logic

Computer message waiting logic can be added to the desired result on any output to the IBM 3270, DCT 1000 or UNISCOPE terminal (text to the screen, text to an auxiliary device, TCS read, TCS search, TCS report address, or TCS backward one block). This is accomplished by the user program performing an OR operation on the special function field with the equate TM#DFSMW (or TM#PFRMW for the IBM 3270), and the equates specified in Table 3-8, if required, for the output type. The following events are collectively defined as computer message wait logic.

- Step 1

  The user program sets the special function byte as described in the preceding paragraph to indicate the usage of computer message wait logic. The user program submits the output to ICAM.

- Step 2

  The RDH builds and sends the computer message waiting command to the terminal. The command lights the MESSAGE WAIT indicator on the display panel and actuates the audible alarm. The RDH retains the output on its line queue until steps 3 and 4 are completed.

- Step 3

  The UNISCOPE operator presses the TRANSMIT key (text from the screen), presses one of the special function keys, or presses the MESSAGE WAIT key. The next poll solicits the input. The DCT 1000 operator presses the transmit key or has the paper tape reader or card reader switch selected. The IBM 3270 operator presses the USM or PA1 key.

- Step 4

  ICAM schedules input or special function key messages to the user program. The MESSAGE WAIT key message is not scheduled to the user program but discarded. DCT 1000 input with a BEL as the first text character is regarded by the RDH as a MESSAGE WAIT key message.

- Step 5

  If the output MCT special function key value was TM#DFSMW, the RDH sends the output to the terminal. If the special function key value was other than this, it sends output only upon receipt of the MESSAGE WAIT key input.

  If computer message wait logic is not in progress for the terminal, then the message wait key input from the terminal is scheduled to the user program. Therefore, the MESSAGE WAIT key (control G on the DCT 1000) can be used as an attention key to the user program.

3.3.1.1.2. UNISCOPE Transmit and Auxiliary Device Transfer Functions

The following information is taken from the UNISCOPE display terminal programmer reference, UP-7807 (current version).

Pressing the TRANSMIT UNPROT DISPL key permits only that data within the unprotected areas on the screen to be transmitted to the processor. The SOE symbol is the only exception because it may be positioned in a protected area. The area transmitted is defined as the unprotected area between the cursor and the SOE symbol nearest to the left of the cursor. In the transmission of data, each time a protected area is reached, the SUB code (octal 032) is inserted as a marker to indicate the omission of protected data. Nonsignificant space suppression is performed from the end of an unprotected field, as well as from the end of a line; if there is more than one unprotected field on a line, nonsignificant space suppression occurs more than once. This allows nonsignificant space suppression within fields, as well as on lines. Nonsignificant space suppression does not occur on the line containing the cursor.

Pressing the TRANSMIT DISPL key on a protected format unit transmits both protected and unprotected data to the processor with the protected fields not marked or identified. Nonsignificant space suppression still occurs.

The DC2 (print) initiates a data transaction between the terminal storage and the auxiliary interface for either an input or an output device, whichever is selected. The format for the exchange is similar to that for exchanges via the communications channel; that is, the area to be transmitted is defined by the SOE character and the cursor, with suppression of nonsignificant spaces and the automatic insertion of cursor returns. The data exchanged between terminal storage and auxiliary interface appears in the screen format. Both protected and unprotected data are transferred under control of the DC2 code. There is no distinction made between protected and unprotected data.

The ESC DC2 (print transparent) initiates auxiliary interface activity between the UNISCOPE terminal storage and the auxiliary interface in the same manner as the DC2 code. The area transmitted is the same as in the DC2 function, but the cursor return characters normally inserted by the logic of the UNISCOPE terminal are not transmitted to the auxiliary interface. This makes the line length of the device on the auxiliary interface independent of the line length of the particular UNISCOPE terminal in use. However, nonsignificant space suppression still occurs.

In addition to the DC2 and ESC DC2, the UTS 400/UTS 4000 allows four other auxiliary device functions:

1. Print form (ESC H) sends to the auxiliary device all of the unprotected characters from SOE (or home position) to the cursor. Field control characters (FCC) and nonsignificant spaces are suppressed.

2. Transfer all (ESC G) sends to the auxiliary device all characters from SOE to cursor. FCC sequences are included and nonsignificant spaces are suppressed.

3. Transfer variable (ESC F) sends to the auxiliary device only the variable (unprotected) characters between the SOE and the cursor. FCC sequences are included and nonsignificant spaces are suppressed.

4. Transfer changed (ESC E) sends to the auxiliary device only the changed characters (or altered fields) between the SOE and the cursor. FCC sequences are included and nonsignificant spaces are suppressed.

The COP/TP must be strapped so that the auxiliary device will space when it receives a DC3 character in the text data. The strap-selectable options are:

1. Ignore all control codes.

2. Space on all control codes.

3. Ignore all control codes but one, which will be detected by a strappable selection, and space on that code.

4. Space on all control codes but one, which will be detected by a strappable selection, and ignore that code.

Option 2, 3, or 4 could be used for selective strapping.

3.3.1.1.3. Output Special Function Field Settings

The meanings and values of the labels used to combine (OR) the values into the special function field are summarized in Table 3-7. The labels are always used rather than the values, since values are volatile and may be changed in various release notices. The labels always remain constant, whereas the equate (EQU) value in the DSECT may vary.

3.3.1.1.4. Soliciting Auxiliary Device Input via Output Commands

When the TCS READ, SEARCH, or REPORT ADDRESS commands result in data being transferred to the UNISCOPE, UTS 400, or UTS 4000 terminals, the following two steps must be taken to make the data available to the user program. When the DCT 500/DCT 524/TTY READ command results in the paper tape reader or the TCS read head being turned on, step 2 must be taken to make the data available to the user program.

NOTE:

The paper tape auxiliary devices of TTYs and DCT 500 terminals, and the magnetic tape auxiliary devices (TCS) of the DCT 524 are capable of the write and read functions only. Therefore, if the auxiliary device field for a TTY, DCT 500, or DCT 524 indicates a write tape device or a read tape device, then a write or a read, respectively, is performed on the auxiliary device, regardless of the content of the special function field.

- Step 1
  The transmit condition is enabled in the UNISCOPE terminal by:

  - the UNISCOPE terminal operator setting the AUTO TRANSMIT switch on the TCS prior to the data transfer from the TCS to the UNISCOPE terminal;

  - the UNISCOPE terminal operator pressing one of the TRANSMIT keys on the terminal, after the data transfer from the TCS to the UNISCOPE terminal; and

  - the user program sending an output text message of the transmit unprotected function (DC1). The text length would be 1. The character would be DC1, which is 11_{16} in EBCDIC. The transmit-all function (ESC DC1) may also be used. The text length would be 2. The characters would be ESC, which is 27_{16} in EBCDIC and DC1. For both types of transmission, the auxiliary device index byte and the special function byte must be zero.

  In the UTS 400 or UTS 4000, the operator should ensure that autotransmit is set in the control page.

- Step 2
  The user program requests the input through the normal input mechanism of the ICAM interface that is being used.

3.3.1.1.5. TCS Search Mode User Messages

The user-supplied text for a TCS search is described in Table 3–8 according to search type.

The RDH expects the characters in Table 3–8 to be the first characters in user-supplied text. (Note that this specifically excludes DICE functions and SOE characters.) The RDH will insert a cursor-to-home sequence (ESCe) and a CAN character ahead of the user-supplied text before sending text to the device. The mode of the search is determined by examining the low order bits of the first character of the user-supplied text, which should be @, A, B, or C. Note also that this text will be placed on the screen at the home position, overwriting anything else there.

Table 3-8. User Message Text for Searching TCS

User Message Text	Search Type
@taaaa or 0taaaa or 'taaaa	Mode search to position the tape, where: @, 0, or ' (grave accent mark) is constant, and: t Is the track address (1 or 2). aaaa Is the address where the tape is to be positioned. If specified as 0000, the tape is rewound.
Ataaaa or 1taaaa or ataaaa	Mode search to position the tape to a particular address and then read one block, where A, 1, or a is constant, and: t Is the track address (1 or 2). aaaa Is the address where the tape is to be positioned.
Btaaaa/c . . . c or 2taaaa/c . . . c or btaaaa/c . . . c	Mode search to position the tape to a particular address, search for a specific character string, and read one block, where B, 2, or b is constant, and: t Is the track address (1 or 2). aaaa Is the block address. c . . . c Is the character string. Up to 16 characters can be specified.
Ct/c . . . c or 3t/c . . . c or ct/c . . . c	Mode search to find the specified character string, where C, 3, or c is constant, and: t Is the track address (1 or 2). c . . . c Is the character string. Up to 16 characters can be specified. The search starts at the present tape position.

3.3.1.2. Interactive Terminal Input

Table 3-9 specifies the types of input that ICAM can submit to the user program. It specifies that ICAM supplies to the user:

- text when required;
- contents of the special function field; and
- contents of the auxiliary device index field.

It also discusses the originator of the input (terminal operator or the user program issuing a read) and the RDH modifications of the input.

Table 3-9. Input Description for Interactive Terminals (Part 1 of 4)

Terminal	Originator of the Input	RDH Modifications of the Input	Is User Supplied Text?	Contents of Auxiliary Device Index Field TM#PDVC	Contents of Special Function Field (1) (TM#PSPEC)
UNISCOPE 100 or 200, UTS 400 or UTS 4000 (all modes)	Operator presses TRANSMIT key. The data on the screen could be keyboard input or a block of data from the TCS obtained by the operator pressing the PRINT key.	Input DICE is applied.	Yes	0	0
	Your program previously issued a TCS read or a mode A, B, or C TCS search.	Input DICE is applied.	Yes	Same value as you specified in the auxiliary device index field when the read/search was issued.	0
	Same as above, with the addition of transparent auxiliary device transfer set in the special function byte when the read/search was issued.	Input DICE is not applied. The RDH removes the SOE cursor sequence and the carriage returns from the text. Thus, the input data is the same as the original output (from your viewpoint).			
	Your program previously issued a TCS report address.	The text supplied to you will be: DICE VT aaaaa where: DICE = X'10010101' which is set coordinate for home position VT = 03_{16} aaaaa = TCS address			
	MESSAGE WAIT key pressed		No	0	TM#PFRMW
	Function key F1, F2, F3, or F4 pressed		No	0	TM#PFKY1
					TM#PFKY2
					TM#PFKY3
					TM#PFKY4
UTS 400, UTS 400 TE and UTS 4000 (native mode)	Firmware returns cursor address (requested by host CUP).		Yes	0	0
	Firmware completed confidence test (operator or host CUP initiated).	(5)	No	0	TM#PPOC
	Firmware returns error log (requested by host CUP).		Yes (3)	0	0

Table 3-9. Input Description for Interactive Terminals (Part 2 of 4)

Terminal	Originator of the Input	RDH Modifications of the Input	Is User Supplied Text?	Contents of Auxiliary Device Index Field TM#PDVC	Contents of Special Function Field (1) (TM#PSPEC)
UTS 400 and UTS 4000 (UTS 400 mode)	Firmware accepts/rejects load program that was to be executed (host CUP initiated load). (6)	No translation performed	Yes (3) (4)	0	0
	Firmware returns dump block (requested by host CUP). (6)	No translation performed	Yes (3) (4)	0	0
	Operator pressed hang-up key. (6)	(5)	No	0	TM#PXEOT
	Operator pressed function keys F5 through F22. (6)		No	0	TM#PFKY5 TM#PFKY6 TM#PFKY7 TM#PFKY8 TM#PFKY9 TM#PFK10 TM#PFK11 TM#PFK12 TM#PFK13 TM#PFK14 TM#PFK15 TM#PFK16 TM#PFK17 TM#PFK18 TM#PFK19 TM#PFK20 TM#PFK21 TM#PFK22
UDS 2000 (UNISCOPE protocol)	Operator presses TRANSMIT key after entering data on screen OR a single record is received from the integral diskette or auxiliary device. (7)	Input DICE is applied, if TERM macro allows.	Yes	0	0
	Multiple records are received from the integral diskette or auxiliary device. Your program had previously issued a TCS read. (7)	Input DICE is applied, if the TERM macro allows.	Yes	Same value you specified in the auxiliary device index field when the TCS read was issued.	0

Table 3-9. Input Description for Interactive Terminals (Part 3 of 4)

Terminal	Originator of the Input	RDH Modifications of the Input	Is User Supplied Text?	Contents of Auxiliary Device Index Field TM#PDVC	Contents of Special Function Field (1) (TM#PSPEC)
DCT 1000	The paper tape reader or the card reader is switched, selected by the terminal operator. The normal pool cycle of ICAM brings in each card image or paper tape block that is in the card hopper or on the paper tape. Thus, you must be prepared for successive inputs.	DICE is applied the same as it is for input from the keyboard. The EM character is removed from the message.	Yes	(2)	0
	First character of input text is BEL.		Yes (7)	0	TM#PFKMW
DCT 500, DCT 524, DCT 475, TTY, UTS 10	Operator does keyboard input. The DCT 500 and DCT 524 can be in TTY or addressed mode.		Yes	0	0
	First character of input text is BEL.		Yes	0	TM#PFKMW
DCT 500, DCT 524, TTY, UTS 10	Operator does paper tape or magnetic tape (TCS) manual input. The DCT 500 and DCT 524 are in TTY mode.		Yes	0	0
	Your program previously issued a read for the input auxiliary device (paper tape reader or TCS read head).	DICE is applied the same as it is for input from the keyboard. Tape device control characters are removed from message.	Yes	Same value as the user specified in the auxiliary device index field when the read was issued.	0
DCT 500, TTY	Interrupt on a DCT 500 or break on a TTY.		No	0	TM#PCBRK

Table 3-9. Input Description for Interactive Terminals (Part 4 of 4)

Terminal	Originator of the Input	RDH Modifications of the Input	Is User Supplied Text?	Contents of Auxiliary Device Index Field TM#PDVC	Contents of Special Function Field (1) (TM#PSPEC)
IBM 3270	Operator presses ENTER key after entering data on the screen. Input DICE is applied if specified in TERM macro.		Yes	0	TM#PENTR
	USM (unsolicited message) key is pressed.		No	0	TM#PFRMW
	FUNCTION keys F1 through F36 are pressed.		Yes	0	TM#PFKY1-9 TM#PFK10-36
	EMPHASIS keys A1 through A10 are pressed.		No	0	TM#PFRMW/ TN#PAK2-10
	TEST key is pressed.		Yes	0	TM#PTREQ
	CLEAR key is pressed.		No	0	TM#PCLER
	Selector pen input is received.		No	0	TM#PLPEN
	Operator ID card input is received.		Yes	0	TM#PMGRN
	Magnetic slot reader input is received.		Yes	0	TM#PMGRA
	No key pressed, and host does an unsolicited read or read modified to a display.			0	TM#PNAD1
	No key pressed, and host does an unsolicited read or read modified to a printer.			0	TM#PNAD2

NOTES:

(1) See 3.3.1.2.1.

(2) The auxiliary input device (card/paper tape reader) must have been defined in the TERM macro AUXn parameter of the CCA network definition. If not, a console message will indicate terminal down because of an address error. The contents of the auxiliary device index field is the auxiliary device index of the AUX parameter (n).

(3) Refer to the UTS 400 programmer reference, UP-8359 (current version).

(4) Refer to the ICAM concepts and facilities, UP-8194 (current version).

(5) If any output messages are awaiting a response (e.g., ACK or THRU) from the terminal when this event is recognized by the RDH, it will mark those messages as unsuccessful (parity error) and will inform CNC by returning the appropriate output MCTs.

(6) Not applicable to UTS 400 text editor.

(7) UDS 2000 operator selects the input device.

3.3.1.2.1. Input Special Function Byte Settings

Table 3-9 describes the labels used to test or set the values into the special function byte of the process file DTF.

3.3.1.2.2. Input Error Notification

There is no input error notification for a STDMCP user. If the error resulted in the final terminal on the line being marked down, there is line-down notification.

Table 3-10 lists the TM#PSPEC status codes that identify input error conditions for the STDMCP user.

Table 3-10. Input Error Notification Status Codes

Terminal	Error Condition		Input Queued	TM#PSPEC
UNISCOPE 100, UNISCOPE 200, DCT 1000, UDS 2000 UTS 400, UTS 4000, IBM 3270	Any input error on which ICAM will① retry and has passed at least one network buffer of data to the DDI user		No	N/A③
	Front end (CA) reported error occurred on last input retry (i.e., response to)		No	N/A
	Invalid message header detected by the RDH on the last input retry			
DCT 500, DCT 524, DCT 475, TTY	Keyboard	Input error (not parity) while opening for input (e.g., input overrun) or during input (e.g., carrier off)	No	N/A
	Tape		No	N/A
	Input parity error①		No	TM#PCRVH
	DCT 524 magnetic tape read error②			0
	Cancel input character received		No	N/A
	Break signal received during input			N/A
	Time-out during keyboard input (e.g., no ETX received within one minute)			N/A
	Time-out during tape input (e.g., no ETX punched on the tape)		No	N/A

NOTES:

① ICAM gives DCT 500, DCT 524, DCT 475, TTY parity error input to the STDMCP user as though it were normal.

② The STDMCP user is notified of a DCT 524 TCS read error as described in note 1. The user can identify the read error condition because the last character of the message is a SYN (32_{16}) character. When a read error occurs on the DCT 524 TCS, the DCT 524 transmits a SYN character and then moves to the next interrecord block gap and waits. The user is responsible for error recovery. (See the DCT 500 programmer reference, UP-7336 (current version).)

③ N/A indicates not applicable or no setting applies.

3.3.1.3. Translation Table Modifications

The EBCDIC-to-ASCII translation table is modified so that selected control characters are translated to the DEL ($7F_{16}$) ASCII character. The control characters, if present in output text, will disrupt the normal terminal hardware logic. The modified entries are SOH (01_{16}), STX (02_{16}), ETX (03_{16}), EOT (37_{16}), SYN (32_{16}), EOB (26_{16}). EM (19_{16}) is also modified for UNISCOPE terminals, but not for UTS 400 or UTS 4000 terminals.

The ASCII-to-EBCDIC translation table is modified so that the DC3 (11_{16}) character is translated to the SP (40_{16}) EBCDIC character.

3.3.1.4. Function Buffering

Certain control characters or control character sequences cause terminal operations that must be allowed to complete before the terminal can receive the next data character. Whether the control characters are user-program supplied or DICE induced, the RDH takes care of inserting the required number of time fill characters. The time fill characters used are:

- NUL (00_{16}) for output to the UNISCOPE, UTS 400, UTS 4000, or DCT 1000 terminals
- NUL or DEL ($7F_{16}$) for output to the DCT 500, DCT 524, DCT 475, or TTY

3.3.2. Interactive Terminals Supported

Each interactive terminal supported by ICAM is discussed in succeeding subsections. The terminals are the UNISCOPE 100, UNISCOPE 200, UTS 400, UTS 4000, DCT 1000, DCT 500, DCT 524, DCT 475, TTY, and IBM 3270. Each discussion includes:

- Reference documents
- Operational considerations
- Special software support

3.3.2.1. UNISCOPE/UTS 400/UTS 4000/DCT 1000 Terminals

The UNISCOPE 100, UNISCOPE 200, UTS 400, UTS 4000, and DCT 1000 terminals are handled by the same remote device handler. They are polled terminals that can be on a line in any mix. . Multiple drops from the line are allowed. Each modulator/demodulator (modem) or each direct connection module (DCM) on the line defines a drop.

A single terminal may be connected directly to the DCM/modem. The rid address must be unique for the line.

Several terminals may be connected to a terminal multiplexer which, in turn, is connected to the DCM/modem. The terminal types can be mixed. The rid of all the terminals on the multiplexer can be common or the terminals can be subdivided into groups where each group has a common rid.

Each single terminal connected directly to the DCM/modem and each group of terminals on the terminal multiplexer is called a poll group. The RDH solicits input from the poll group with a single traffic poll whose address has the rid of the poll group. The sid is general.

Each poll group is periodically polled at a rate defined by the PINTV keyword parameter of the TERM macroinstruction used in the network generation. The poll rate may be different for each poll group.

When a terminal is marked down by ICAM because of unrecoverable errors, a terminal down message giving the reason and terminal name is sent to the system console. ICAM stops sending text to the downed terminal.

Polling continues even though a terminal is marked down. The poll rate remains unchanged if at least one terminal in the poll group is still up. If all terminals in the poll group are down, and at least one terminal in another poll group is up (the line is not down), the poll rate is slowed to once each minute. The operator can still try input from the terminal, but must be aware of this slow poll rate. If the input is successful, a terminal up message is sent to the system console giving the terminal name. Sending text to the terminal is continued and the poll rate is changed back to the value defined in the network generation.

The most efficient communication is achieved by having one poll group per line. All the terminals on a line would be connected to a single terminal multiplexer. Each terminal would have the same rid in its address.

3.3.2.1.1. Terminal Multiplexer and Direct Connection Module (DCM)

Refer to the following manuals for a more detailed description of the terminal multiplexer and the direct connection module (DCM).

- Direct connection module functional description, UP-7932 (current version)
- Terminal multiplexer functional description, UP-7916 (current version)

3.3.2.1.2. Utilization of the RDH Line Queue

In certain ICAM configurations, the RDH maintains its own output queue on a line basis. The line queue can contain one output per terminal. This queue is not the same as the message queues generated by the LOW, MEDIUM, and HIGH operands of the LINE or TERM macroinstruction.

If an output to an auxiliary device connected to a UNISCOPE, UTS 400, UTS 4000, or DCT 1000 terminal causes a local busy (COP printing, etc), the output is retained on the RDH line queue until the local busy is completed. The RDH can send other output from the queue to another terminal during the local busy.

To ensure the proper utilization of the RDH line queue, a message queue must be defined for each terminal. The LOW keyword parameter of the TERM macroinstruction must be used:

LABEL	△OPERATION△	OPERAND
name	TERM	LOW=MAIN

You can also use priority queueing with the TERM macroinstruction by specifying the MEDIUM or HIGH operand.

NOTE:

Utilization of the RDH line queue also increases efficiency in the line protocol when there are several terminals in a poll group.

3.3.2.1.3. UNISCOPE Terminal Considerations

The UNISCOPE display terminal is keyboard operated and has a message display screen. It can support the following auxiliary devices:

- Tape Cassette System (TCS)
- Communications Output Printer
- 800 Terminal Printer (TP)
- 0786 Printer

The UNISCOPE auxiliary interface permits up to 8 auxiliary devices and up to 12 device addresses that ICAM can support.

3.3.2.1.3.1. Reference Documents

Refer to the current versions of the following manuals for a more detailed description of the UNISCOPE display terminal and its auxiliary devices:

- UNISCOPE display terminal concept and applications, UP-8155
- UNISCOPE display terminal general description, UP–7701
- UNISCOPE display terminal operator reference, UP-7788
- UNISCOPE display terminal programmer reference, UP-7807
- UNISCOPE display terminal auxiliary interface, UP-7855
- UNISCOPE display terminal communications output printer functional description, UP-7939
- Model 610 tape cassette system component description, UP-8012
- Model 800 terminal printer functional description, UP-8013

3.3.2.1.3.2. Operational Considerations

See the UNISCOPE display terminal operator reference, UP-7788 (current version), for operational instructions.

3.3.2.1.4. DCT 1000 Terminal Considerations

The DCT 1000 terminal is a fully buffered, 30-characters-per-second incremental printer that can be expanded to include a keyboard, a card reader, a card punch, a paper tape reader/punch, and an auxiliary printer.

3.3.2.1.4.1. Reference Documents

Refer to the current versions of the following manuals for a more detailed description of the DCT 1000:

- DCT 1000 data communications terminal general description, UP-7782
- DCT 1000 data communications terminal operator reference, UP-7828
- DCT 1000 data communications terminal programmer reference, UP-7859

3.3.2.1.4.2. Operational Considerations

When using the DCT 1000, note the following restrictions:

- The output selection switches for auxiliary devices (i.e., paper tape punch, printer) must be in the OFF position when the DCT 1000 is online to ICAM. The switches are used only to perform offline functions.

- Only one input selection switch is allowed on at any one time. The DCT 1000 operator must ensure that the proper input device is selected at all times.

- The ICAM user program cannot select the input device. If your program tries to select an input device that is different from the DCT 1000 switch settings, no input is received from the device selected by your program. Input is received on the next poll of the DCT 1000 from the device selected by the switch settings.

- The operator must be aware of the main storage lock in the DCT 1000. Any time an input device has access to the DCT 1000 main storage, no output can be sent to the terminal. If the operator inadvertently activates a key that locks main storage while waiting for output, the output is not sent to the terminal. The operator must press the TRANSMIT key to clear the main storage lock. The input is scheduled to the user program and the output is then sent to the DCT 1000.

- After the user-supplied output has been modified by the RDH DICE processing, its length must not exceed 160 characters. If it does, it will be truncated by the RDH.

- For text from the user to the DCT 1000 card punch, the RDH first replaces DICE with the proper character control characters and then pads the text to a card image boundary. There are four cases to consider. They are:

  1. The DICE-modified user text is less than 80 characters. Padding is done to a full card image (80 characters). One card is punched.

  2. The DICE-modified user text is 80 characters. No padding is done. One card is punched.

  3. The DICE-modified user text is greater than 80 characters and less than 160 characters. Padding is done to a full second card image (total of 160 text characters for the two card images). Two cards are punched.

  4. The DICE-modified user text is 160 characters. No padding is done. Two cards are punched.

  NOTE:

  Sending one card at a time is nearly as fast as sending two. The card punch and the DCT 1000 each have a buffer. This means that while a card is being punched, the DCT 1000 buffer can be filled with the next card image.

3.3.2.1.5. UTS 400 Terminal Considerations

The UTS 400 is a general purpose, microprocessor-based remote display terminal system used for interactive data communications with a central processing system (host processor). The terminal is available in two different configurations: as a master station with zero, one, or two slave stations, and as a controller unit with a minimum of one and up to six slave stations. The UTS 400 can support the following auxiliary devices:

- Model 800 terminal printer
- Communications output printer
- Tape cassette system
- Diskette subsystem
- 0774 printer
- 0786 printer
- 0791 correspondence quality printer

The UTS 400 can be intermixed with other Sperry Univac terminals, such as the DCT 1000 and UNISCOPE 100/200 Display Terminals.

3.3.2.1.5.1. Reference Documents

Refer to the current versions of the following documents for more information concerning the UTS 400:

- System description, UP-8357
- Programmer reference, UP-8359
- Operator reference, UP-8358

3.3.2.1.5.2. User Considerations

The following considerations apply to the use of the UTS 400 Universal Terminal System:

- Specifying UTS 400 System usage

  You specify UTS 400 terminal usage in the LINE and TERM macroinstructions of your network definition as follows:

```
(label) LINE DEVICE=(UNISCOPE)
(label) TERM FEATURES=(U400,screen size,SBT,{CP})
                                            {PR}
```

  The UTS 400 may be configured with the screen bypass feature (SBT), in UNISCOPE mode with 15 protected fields per line (PR), or 80 protected fields per line plus function keys 5 through 22 (CP). UTS 400 configurations are described in the ICAM network definitions and operations user guide, UP-8947 (current version).

- Screen Bypass Usage

  The maximum number of terminals that can be connected to a UTS 400 controller is seven (six slaves and one screen bypass feature device).

3.3.2.1.5.3. How to Control the Display Screen on a UTS 400 Terminal

Normally, you control the display screen of a UTS 400 terminal from the keyboard of the terminal. You can also perform most of these functions from your program in the central processor by placing screen control and field control characters (FCCs) in the messages you send to the UTS 400. Table 3-11 lists the screen control characters used in your messages to control a UTS 400 display screen. Figure 3-1 shows how a display screen may be addressed by coordinates. This figure also shows the values you can place in messages to move the cursor to a specific location. Then you may begin writing or set a field control character.

To illustrate the use of screen control codes in your program, if you want to begin a message on a display at line 2 (!), column 5 ($), prefix your text message as follows:

```
ESC VT ! $ SI....text....
```

Or if you want to start your message at line 1, column 1 (cursor to home), precede your message with ESC e as follows:

```
ESC e......text....
```

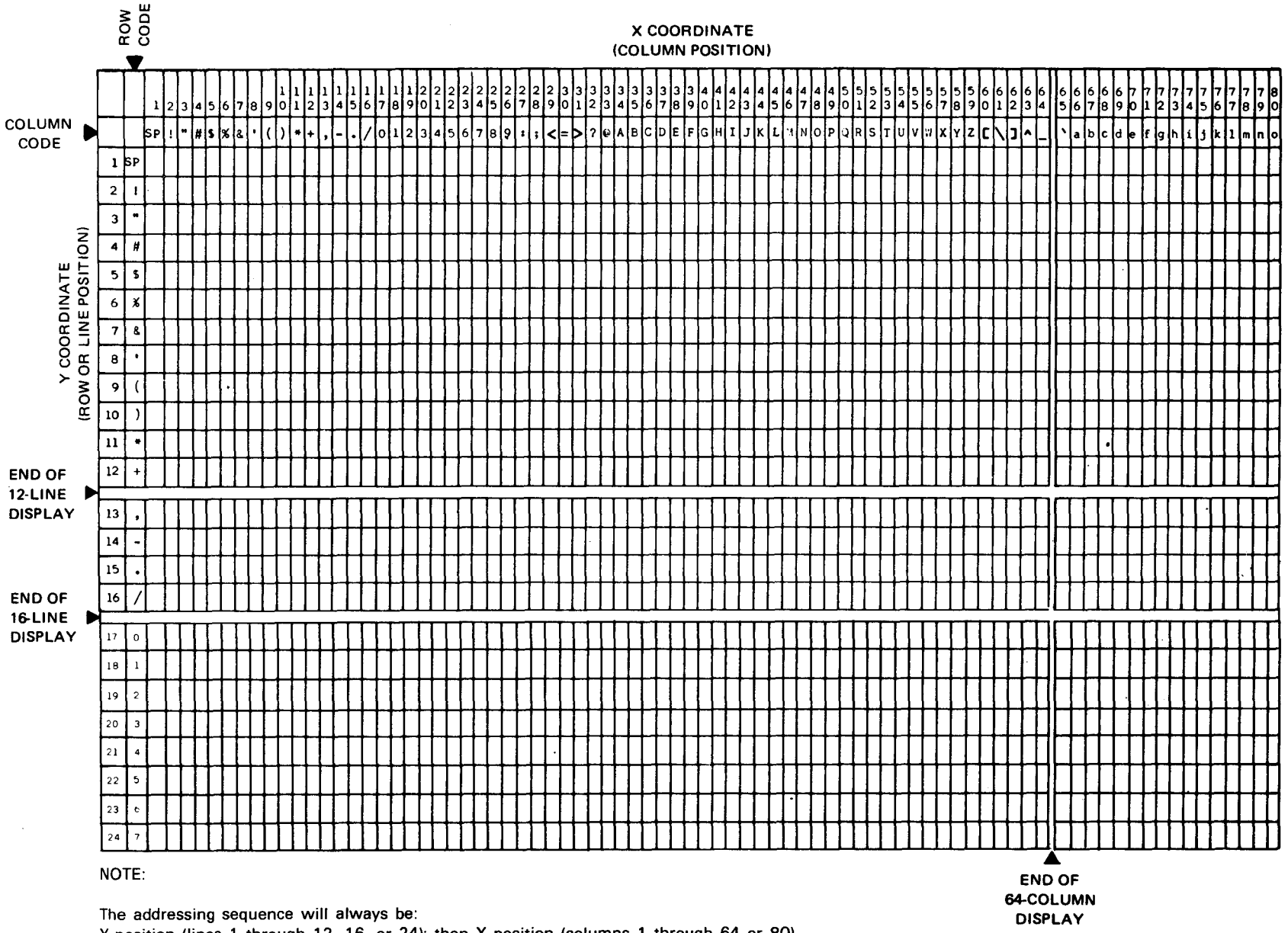
The code sequence we just used and others are described in Table 3-11. Both examples control the position of the cursor only. However, the UTS 400 also lets you assign field control characters. Some functions may be defined through the use of DICE.

Table 3-11. UTS 400/UTS 4000 Screen Control Codes

Function	Code/Sequence
Cursor positioning	ESC VT Y X SI
SOE position	ESC VT Y X NUL SI
Start of entry (SOE)	RS
Cursor return (new line)	CR
Cursor to home	ESC e
Send cursor address*	ESC T
Erase unprotected data	ESC a
Erase to end of line	ESC b
Erase to end of field	ESC K
Erase display	ESC M
Erase character (space)	SP
Delete in line	ESC c
Delete in display	ESC C
Delete line	ESC k
Insert in line	ESC d
Insert in display	ESC D
Insert line	ESC j
Line duplication*	ESC y
Scan left	ESC g
Scan right	ESC h
Scan down	ESC i
Scan up	ESC f
Forward tab	HT
Tab stop set	ESC HT
Backward tab*	ESC z
Control page*	ESC o

Function	Code/Sequence
Transfer changed*	ESC E
Transfer variable*	ESC F
Transfer all*	ESC G
Print form*	ESC H
Print	DC2
Print transparent	ESC DC2
Transmit variable	DC1
Transmit all	ESC DC1
Transmit changed*	ESC t
Clear changed*	ESC u
Call error log*	ESC P
Clear error log*	ESC R
Initiate confidence test*	ESC O
Blinking start marker	FS
Blinking end marker	GS
Lock keyboard	DC4 or ESC DC4
Shift in**	SI
Shift out**	SO
Line feed	LF
Form feed	FF
FCC character sequence*	US R C M N
FCC character clear*	ESC w

* UTS 400/UTS 4000 only (not supported on UNISCOPE terminals).
** Meaning depends upon FCC/PROTECT switch setting.



NOTE:

The addressing sequence will always be:
Y position (lines 1 through 12, 16, or 24); then X position (columns 1 through 64 or 80).

Figure 3-1. Screen Coordinate Cursor Addressing for Display

Field control characters assign attributes such as intensity, tabulation, protection, and justification to one or more fields in the display. You set field control characters at the keyboard by placing the cursor at the desired position and pressing the UPPER FUNCTION and FCC GENERATE keys followed by the attributes you want. When all of the field control characters are entered and stored in the terminal, activate them by pressing the FCC REENABLE key.

The following steps summarize how to set field control characters at the keyboard of a UTS 400 terminal. However, you should obtain a copy of the current version of the UTS 400 operator reference guide, UP-8358, for complete details.

1. Place cursor at first position of the desired field.

2. Press FCC GENERATE and UPPER FUNCTION simultaneously.

3. Define desired FCC functions by sequentially entering four code characters as defined:

Entry Sequence	FCC Function	Code Characters	Meaning of Entry
1	Intensity	N (or space)	Normal intensity (high)
		L	Low intensity
		B	Blink
		O	Off
2	Tab stop	T	Tab (FCC acts as tab stop)
		S	FCC is skipped in tab operation.
		(or space)	
3	Field restrictions	P	Protected field (no entry allowed)
		A	Alphabetic entries only
		N	Numeric entries only
		U	Any entry allowed
		(or space)	
4	Justification	R	Right justification (of all data entered)
		Space	Normal placement of data

4. Release FCC to storage by pressing the space bar or any data key. All FCC attributes are operational *except protected field and right justification.*

When all desired FCCs are entered and released to storage, press the FCC REENABLE key to activate the protected-field and right-justification functions.

You may also initiate field control characters from your program by placing FCC sequences into messages you send to the UTS 400 terminal. Each sequence is five bytes long as follows:

US y x m n

where:

US
Is a unit separator character used to start an FCC sequence.

y
Is the row (line) in which the FCC is located.

x
Is the column in which the FCC is located.

m and n
Specify the attributes of the FCC. Table 3-12 lists the m values you can specify. Table 3-13 lists n values you can specify.

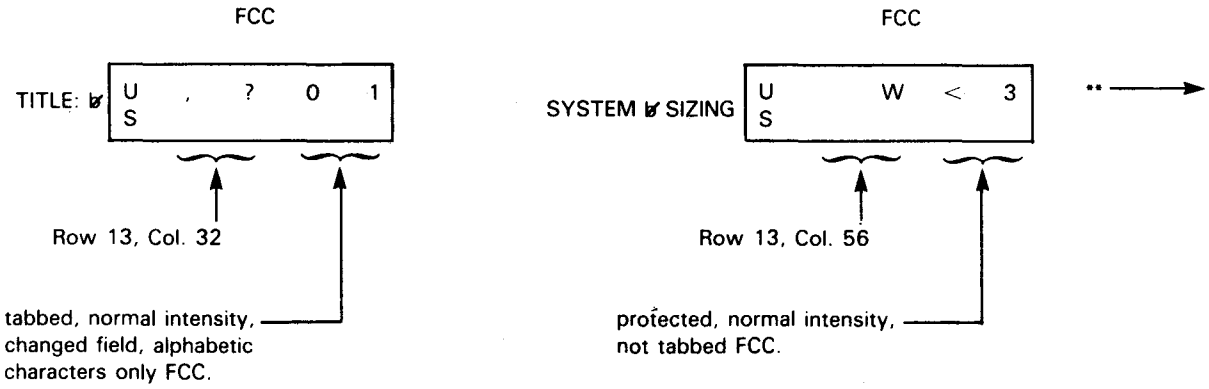
Table 3-12. ASCII Characters Used as m in the UTS 400 FCC Sequence

ASCII Character	Octal Code	Hexadecimal Code	Field Characteristics
0	60	30	Tab stop, normal intensity, changed field
1	61	31	Tab stop, display off (no intensity), changed field
2	62	32	Tab stop, low intensity, changed field
3	63	33	Tab stop, blinking display, changed field
4	64	34	Tab stop, normal intensity
5	65	35	Tab stop, display off (no intensity)
6	66	36	Tab stop, low intensity
7	67	37	Tab stop, blinking display
8	70	38	Not tab stop, normal intensity, changed field
9	71	39	Not tab stop, display off (no intensity), changed field
:	72	3A	Not tab stop, low intensity, changed field
;	73	3B	Not tab stop, blinking display, changed field
<	74	3C	Not tab stop, normal intensity
=	75	3D	Not tab stop, display off (no intensity)
>	76	3E	Not tab stop, low intensity
?	77	3F	Not tab stop, blinking display

Table 3-13. ASCII Characters Used as n in the UTS 400 FCC Sequence

ASCII Character	Octal Code	Hexadecimal Code	Field Characteristics
0	60	30	Any input
1	61	31	Alpha only
2	62	32	Numeric only
3	63	33	Protected (no entries and no changes)
4	64	34	Any input, right-justified
5	65	35	Alpha only, right-justified
6	66	36	Numeric only, right-justified

The following is an example of using field control characters to set up your UTS 400 display screen.



In this example, the first FCC character lets the terminal user tab to the first FCC identified location and enter alphabetic data. The data typed displays with normal intensity (brightness).

The terminal user types in the words SYSTEM SIZING.

The second FCC character protects the field beginning with two asterisks; that is, the terminal user cannot type over or change the asterisks. The asterisks are in normal intensity, and they are not tabbed.

3.3.2.1.6. UTS 400 Text Editor (TE) Considerations

The TE is a special purpose, microprocessor-based remote terminal system designed for text editing in the printing and publishing industry. The TE supports the following auxiliary devices:

- Model 800 terminal printer
- Communications output printer
- Tape cassette system
- Diskette subsystem
- 0786 printer

Refer to the UTS 400 text editor system description, UP-8411 (current version) for further information.

3.3.2.1.7. UTS 4000 Terminal Considerations

The SPERRY UNIVAC Universal Terminal System 4000 (UTS 4000) is a remote terminal system used for interactive data communications with a central processor. As shown in Figure 3-2, the UTS 4000 terminal system consists of:

- SPERRY UNIVAC Universal Terminal System 20 (UTS 20) Single Station, which is not programmable.
- SPERRY UNIVAC Universal Terminal System 4020 (UTS 4020) Cluster Controller, supporting up to 12 UTS 20W workstations.
- SPERRY UNIVAC Universal Terminal System 40 (UTS 40) Single Station, which is programmable.

Peripherals available on the UTS 4000 system and their applicability are listed in Table 3-14; Figure 3-2 shows a UTS 4000 system making use of all of these terminals.

The current versions of the following documents apply to the UTS 4000 devices supported by ICAM.

- Universal Terminal System 20 (UTS 20) Single Station

  System Description, UP-9134
  Operators Guide, UP-9135
  System Reference, UP-9136

- Universal Terminal System 40 (UTS 40) Single Station

  System Description, UP-9141
  Operators Guide, UP-9142
  System Reference, UP-9143

- Universal Terminal System 4020 (UTS 4020) Cluster Controller

  System Description, UP-9149
  System Reference, UP-9150
  Universal Terminal System 20 (UTS 20W) Workstation Operators Guide, UP-9156

Table 3-14. UTS 4000 Peripheral Device Support

Peripheral Name	UTS 20S/ UTS 20W	UTS 40 (Single Station)	UTS 4020 Cluster Controller
F3389 Magnetic Stripe Reader	Y	Y	N/A
0797 Printer Subsystem	Y	Y	Y
0798 Printer Subsystem	Y	Y	Y
0791 Correspondence Quality Printer Subsystem	N	Y	Y
8406 Double Sided Diskette Subsystem	N	Y	Y

LEGEND:

Y	indicates that the device is supported.
N	indicates that the device is not supported.
UTS 20S	specifies a UTS 20 single station.
UTS 20W	specifies a UTS 20 workstation.

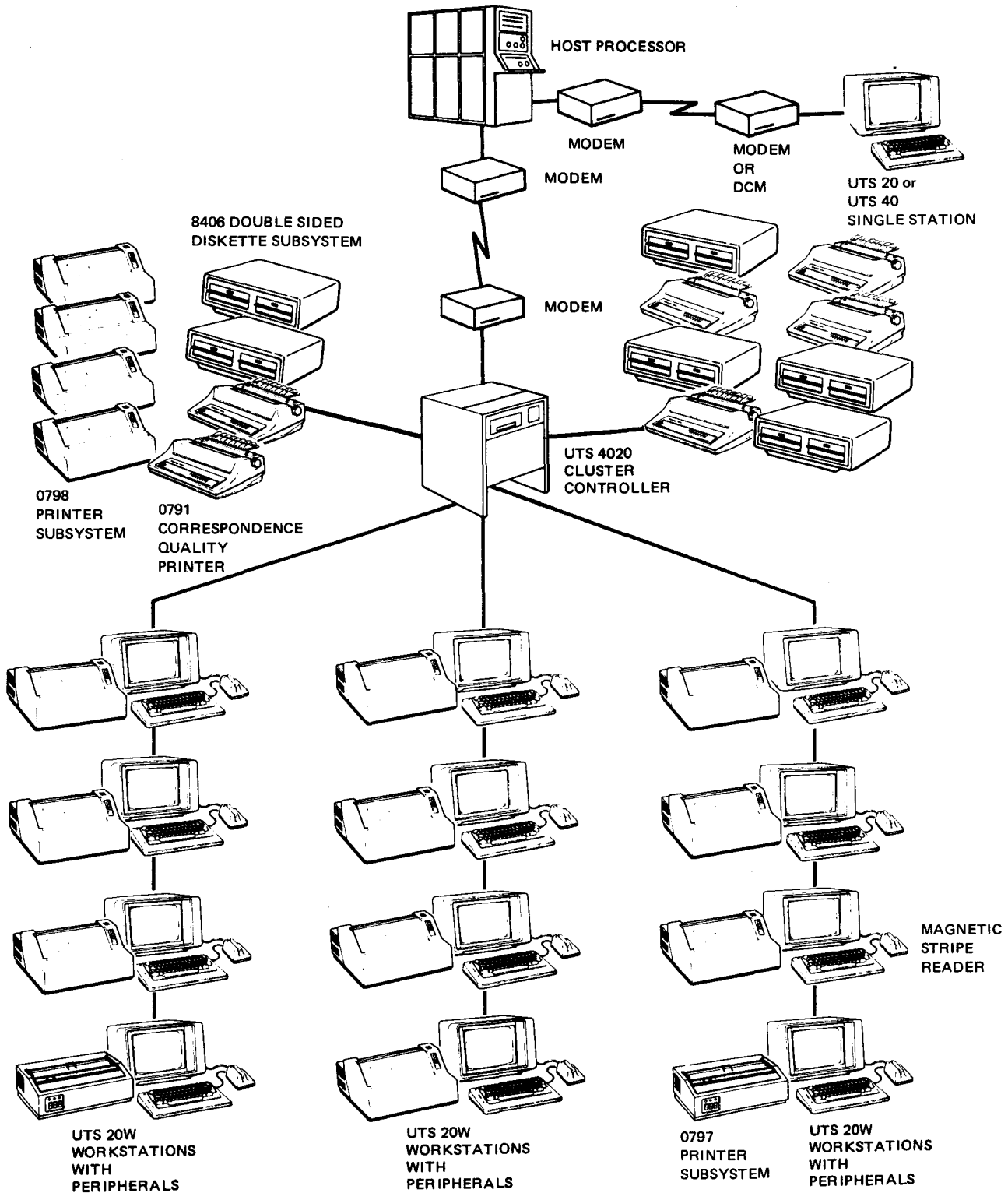


Figure 3-2. ICAM Supported UTS 4000 System with a UTS 4020 Cluster Controller and UTS 20 or UTS 40 Single Station

3.3.2.1.7.1. User Considerations

The following considerations apply to UTS 4000 terminals:

- You define the use of the UTS 4000 terminal in the ICAM LINE macroinstruction.

  If you are using a UTS 4000 as a remote workstation, specify the DEVICE operand as:

  ```
  label LINE DEVICE=(RWS), ...
  ```

  If you are using a UTS 4000 local workstation, specify the DEVICE operand as:

  ```
  label LINE DEVICE=(LWS), ...
  ```

  If you are using the UTS 4000 as a terminal (i.e., you are not using a UTS 4000 workstation), specify the DEVICE operand as:

  ```
  label LINE DEVICE=(UNISCOPE), ...
  ```

- You define the features supported on each UTS 4000 model on the ICAM TERM macroinstruction as follows:

  - Cluster controller

    Some UTS 4000 models attach to a cluster controller. If you are using one of these, specify the CC operand; for example:

    ```
    label TERM DEVICE=(U20,1920,,CC) ...
    ```

  - Screen bypass/dual screen

    Some UTS 4000 models include a screen bypass feature (also known as dual screen). This feature, which is really a dual screen memory in the terminal, allows the host processor to communicate with peripheral devices without affecting data (or entry of data) on the displayed screen.

    If you use this feature, you must define two TERM macroinstructions. Each one addresses a separate screen memory in the terminal. For example,

    ```
    label TERM DEVICE=(U20,1920,SBT),ADDR=(35,51) ...
          TERM DEVICE=(U20,1920,SBT),ADDR=(35,52) ...
    ```

    Note that if your terminal includes the screen bypass feature, you must define both screens whether you use them or not.

- PRIMARY/SECONDARY

  If you are using a UTS 4000 remote workstation, and you have the screen bypass/dual screen feature, you must define the primary and secondary screens using the PRIMARY and SECONDARY operands. The SBT operand has no meaning for remote workstation. For example,

```
label TERM DEVICE=(U20,1920),ADDR=(35,51),PRIMARY ...
      TERM DEVICE=(U20,1920),ADDR=(35,52),SECONDARY ...
```

- Additional Device Identifiers

  For the UTS 4000 only, the range of auxiliary device identifiers (DIDs) you can specify is expanded to 92. That is, you may specify different DIDs than for the UNISCOPE or UTS 400 terminals, up to a maximum of 12 per terminal. Specify DIDs in the AUXn operand of the TERM macroinstruction. Values from hexadecimal 20 to hexadecimal 7E (except 70, 71, and 72) are allowed. The general DID remains hexadecimal 70.

- FCC Expansion and Screen Size

  The UTS 4000 permits up to 80 field control characters plus 80 characters of emphasis on each line of the display screen. If the entire emphasis and field control character capability is used, it could result in a message 16,520 bytes long. However, the UTS 4000 does not transmit or receive messages greater than 4096 bytes. Messages exceeding 4096 bytes are sent in segments.

  When the UTS 4000 sends segmented messages to the host processor, it places an ETB control character into each segment that does not terminate a message. The ETB control character is always the last data character in the segment.

  The ICAM remote device handler does not recognize this segmentation and passes each segment to your program as a complete message. Therefore, your program must be prepared to recognize the segmented messages it receives.

  In the case of remote workstations, your program never receives the ETB characters, and an entire message is received as one input.

  Except for a remote workstation, when your program sends a long message to the UTS 4000, it is responsible for dividing it into segments that do not exceed 4096 bytes. The last character of each segment must be ETB except for the last segment in the message.

  When your program sends a large message to a remote workstation, it need not supply ETB characters.

- FCC Start Field Sequence

  A 3-byte sequence called FCC start field (EM m n) enables your program to specify a field control character at the current location of the cursor on a UTS 4000 display.

  The m and n values are the same as in the 5-byte sequence used in your program to set field control characters. You don't need to specify the coordinates of the display location. The start field form of the FCC can only be used in messages your program sends to UTS 4000 terminals. All field control characters sent to your program *from* the UTS 4000 terminal use the 5-character sequence US R C m n.

  See Table 3–11 for a description of these code sequences.

- Data Throttling

  Data throttling prevents the premature overlaying of messages sent to a UTS 4000 terminal for processing. Except for the UTS 20S terminal, the UTS 4000 supports terminal resident user programs that are not visible to the host processor. These UTS 4000 resident programs help process messages sent from the host processor. Therefore, a UTS 4000 resident program's receiving data-area may be overlayed by subsequent messages if it is not able to process each message fast enough and ICAM has more than one message queued for output. This problem is prevented within the UTS 4000 terminal by a technique called data throttling.

  During the periods that your UTS 4000 resident program is processing a message, the UTS 4000 supervisor responds to polls from the central processor with a busy status. Each time your UTS 4000 resident program finishes processing a message and requests a new one, the UTS 4000 supervisor responds to a poll that it is ready to receive. This tells the ICAM remote device handler that the terminal is not busy and it can send the next message.

- 8406 Double-Sided Diskette Subsystem Support

  ICAM supports the 8406 double-sided diskette subsystem to provide offline random access storage on flexible diskettes for the UTS 4020 cluster controller and the UTS 40 single station. This desk top device writes onto diskette or reads from diskette upon command from the UTS 4020, UTS 40, or your program.

  Tape cassette format is used to write to and read from the central processor to the diskette. The following restrictions apply when you use the 8406 diskette:

  - The maximum number of bytes per track is 8128.
  - The end of disk address is 17352.
  - The diskette subsystem is not supported on remote workstations.

- Magnetic Stripe Reader Support

The magnetic stripe reader is a read-only device used to enter prerecorded data from the magnetic stripe on bank cards or similar media. Data is sent to the central processor as from a UTS 20W workstation, a UTS 20S single station, or a UTS 40, except that a DEL character precedes the data. If you write a program in one of your UTS 4000 terminals to handle the magnetic stripe reader, it must be prepared to handle the DEL character, as well as the format-dependent start and stop sentinels.

The following considerations also apply:

- When the autotransmit option is used for the reader, the magnetic stripe reader data bypasses the terminal resident user program. If data throttling is in effect, the magnetic stripe data presented to the host as input causes the output to be treated as an error.

- For the UTS 20 display only, the autotransmit option always causes suppression of input display from the reader. FCC selection may govern UTS 40 display.

- The magnetic stripe reader does not require a device address because the data is treated as a keyboard entry. Reader input cannot be activated by the central processor; operator intervention is always required.

- Field Control Character Support for the UTS 20 and UTS 40 Single Stations

UTS 4000 single-station support of field control characters differs slightly from that of the UTS 400 terminals. The support provided is shown in Tables 3-15 and 3-16. A reverse intensity attribute is added to some m-field characteristics for both of the UTS 4000 single stations. However, the UTS 20 does not support all of the m-field characteristics available on the UTS 40.

Table 3-15. ASCII Characters Used as m in FCC Sequences for UTS 20 and UTS 40 Terminals (Part 1 of 2)

ASCII Character	Octal Code	Hexadecimal Code	Field Characteristics	UTS 20**	UTS 40**
0	60	30	Tab stop, normal intensity, changed field*	Y	Y
1	61	31	Tab stop, display off (no intensity), changed field*	N	Y
2	62	32	Tab stop, low/reverse intensity, changed field*	Y	Y
3	63	33	Tab stop, blinking display, changed field*	Y	Y
4	64	34	Tab stop, normal intensity	Y	Y
5	65	35	Tab stop, display off (no intensity)	N	Y
6	66	36	Tab stop, low/reverse intensity	Y	Y
7	67	37	Tab stop, blinking display	Y	Y

Table 3-15. ASCII Characters Used as m in FCC Sequences for UTS 20 and UTS 40 Terminals (Part 2 of 2)

ASCII Character	Octal Code	Hexadecimal Code	Field Characteristics	UTS 20**	UTS 40**
8	70	38	Not tab stop, normal intensity, changed field*	Y	Y
9	71	39	Not tab stop, display off (no intensity), changed field*	N	Y
:	72	3A	Not tab stop, low/reverse intensity, changed field*	Y	Y
;	73	3B	Not tab stop, blinking display, changed field*	Y	Y
<	74	3C	Not tab stop, normal intensity	Y	Y
-	75	3D	Not tab stop, display off (no intensity)	N	Y
>	76	3E	Not tab stop, low/reverse intensity	Y	Y
?	77	3F	Not stop, blinking display	Y	Y

* When the host processor generates an FCC, the changed-field designator is cleared. However, the host processor generates individual FCCs with the changed-field designator set; this capability may be used for selective transfer or transmission of fields that were not changed by the UTS operator. By sending an ESC u code to the terminal in a text message, the host processor can clear the changed-field designators in all FCCs without regenerating each FCC and without altering the data within the fields.

** Y means supported; N means not supported.

Table 3-16. ASCII Characters Used as n in FCC Sequences for UTS 20 and UTS 40 Terminals

ASCII Character	Octal Code	Hexadecimal Code	Field Characteristics
0	60	30	Any input
1	61	31	Alpha only
2	62	32	Numeric only
3	63	33	Protected (no entries and no changes)
4	64	34	Any input, right-justified
5	65	35	Alpha only, right-justified
6	66	36	Numeric only, right-justified

3.3.2.1.8. IBM 3270 System Support

ICAM provides the IBM 3270 remote device handler to support the IBM 3270 terminal system (3270). The system includes the 3271 control unit, 3277 display station, and the 3284 and 3286 printers.

The 3270 system can be connected point-to-point or it can be multidropped on a line. Each drop on a line requires a modulator/demodulator (modem) or a direct connection module (DCM); only one 3271 can be connected to each modem or direct connection module. For more information on the direct connection module, refer to the current version of the direct connect module functional description, UP-7932. Up to 32 display stations or printers can be connected to each 3271 control unit.

The 3270 remote device handler solicits input from the 3271 control unit with a general poll. Each 3271 control unit is polled at a rate you specify in the PINTV operand of the TERM macroinstruction, when you define your ICAM network. (See the current version of the ICAM network definition and operations user guide, UP-8947 for details.)

A line servicing 3270 terminal systems is marked down when an unrecoverable hardware error status is received. A terminal is marked down because:

- An unrecoverable error occurs.

  The terminal is retried the number of times specified in the RETRY operand of the LINE macroinstruction.

- Abnormal sense/status occurs.

  The terminal is retried the number of times specified in the RETRY operand of the LINE macroinstruction.

- A busy condition is detected by the remote device handler (WACK received).

  The remote device handler determines that printing will not complete successfully or no further response is received from the terminal for 75 seconds.

When a terminal line is marked down, a message describing the reason for the action is sent to the system console. Also, output is no longer sent to the terminal. Polling continues even though a terminal is marked down. The polling rate remains unchanged if at least one terminal in the polling group is still up. If all terminals in the polling group are down and at least one terminal in another polling group is up, the polling rate is slowed to one per minute. If input is successful, a terminal-up message giving the terminal name is sent to the system console. At this time, the terminal is marked up and polling continues at the normal rate. Output, if any, is transmitted.

When sending output a 3-byte sequence (ESC command WCC) is required. If your program supplies the sequence, the command may be a WRITE or ERASE/WRITE with WCC set to the appropriate function.

If your program does not supply the command sequence, the remote device handler supplies a WRITE command and the appropriate WCC character (hexadecimal C3 for a display or hexadecimal C8 for an unformatted message to a printer).

In addition, the remote device handler always inserts an IC character immediately before the end-of-text character (ETX) if the message is addressed to a display. If the message is addressed to a printer, an EM character is inserted before the ETX.

If a formatted message is addressed to a printer, the remote device handler supplies the ETX character only.

Because each IBM 3277 display station and 3284/3286 printer is designated as a primary device, your program can address each one by the terminal name in the TERM macro. These devices support EBCDIC transmission code only.

If an IBM 3284/3286 printer is turned off while a message is being sent, the operator should wait at least 75 seconds before turning the printer on. Otherwise, the RDH is unaware that the printer didn't receive the complete message and, therefore, will not retransmit the message.

3.3.2.1.9. IBM 3270 Emulator

The 3270 Emulator provides a way to connect the OS/3 System 80 to an IBM host system. It allows System 80 workstation users to access applications and IBM program products running on an IBM host. To do this, System 80 operates in emulation mode; it pretends to be a 3270 terminal system. The 3270 Emulator acts as an IBM 3271 control unit, providing the following functions:

1. Communicates between System 80 and the IBM host system using binary synchronous, medium speed, half-duplex communication facilities. It supports EBCDIC line code at speeds up to 9600 bps on switched and leased lines.

2. The System 80–3270 Emulator functions as a slave to the IBM host, responding to polls from the IBM system.

3. It supports two types of user interfaces:

   - User programs can communicate with the IBM host through the 3270 Emulator. User programs can be written in BAL using dedicated networks or using global networks with static or dynamic sessions. User programs can be written in COBOL using dedicated networks or using global networks with static sessions.

   - A System 80 local workstation functions as a 3277 Model 2 display station. It supports formatted and unformatted screens, protected fields, field attributes, function keys, screen format control, and up to 32 dynamic (terminal to terminal) sessions. Workstations can interactively update files, run jobs on an IBM processor, and run normal OS/3 jobs and ICAM programs.

4. It interprets 3270 commands and orders embedded in the message text. The control characters embedded in the text can, at your option, be passed, stripped, or converted to the UTS 4000 local workstation formats by the 3270 Emulator.

5. It supports IBM Transparent Monitor Mode, enabling the connection of other binary synchronous communication units on the same communication line.

Figure 3-3 shows how the 3270 Emulator operates.

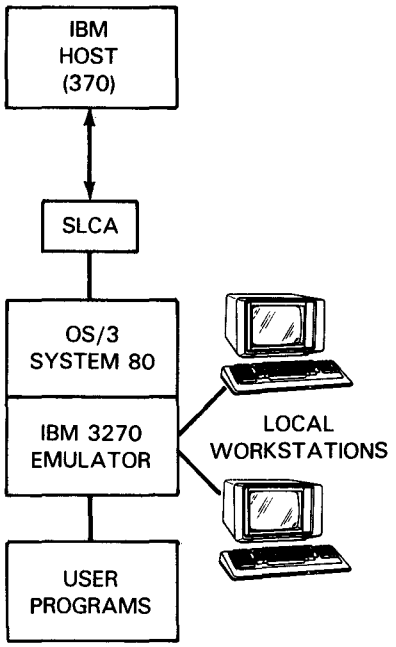


Figure 3-3. The IBM 3270 Emulator

3.3.2.1.9.1. Hardware and Software Support

The 3270 Emulator needs the support of:

- a Series 90 processor with communication adaptor; or
- a System 80 processor with a single-line communication adaptor, feature number F2788-02 or F2788-03.

The 3270 Emulator is part of the ICAM terminal support package. If you're using local workstations, you must generate local workstation terminal support and the global network facility in ICAM. The 3270 Emulator adds approximately 14K bytes to an ICAM generation.

3.3.2.1.9.2. Operational Considerations

When you use the 3270 Emulator, the following restrictions apply.

- 3270 Model 1 control and 3277 Model 1 display (480-character screen size)
- Selector light pen
- Operator identification card reader
- Printers
- IBM program attention and program function key considerations:
  - IBM program attention keys PA1-PA4 are mapped by local workstation keys F1-F4.
  - IBM PF1-PF18 are mapped by local workstation keys F5-F22 (the IBM 3277 terminal only supports keys PF1-PF12).
  - IBM 3277 PF19-24 aren't supported on the local workstation, but a user program can use them.
- Test request key
- ASCII support

Unsupported IBM commands, orders, and functions are:

- Copy command feature
- Read buffer (diagnostic) command
- Diagnostic read and write commands
- Sound alarm (write control character (WCC))

- Nondisplay fields (attribute byte (ATB))
- Screen wrapping

When the 3270 Emulator communicate with the IBM host, note the following:

- The 3270 Emulator manipulates all data in EBCDIC. Data passed between the local workstation and the 3270 Emulator is translated by the hardware to ASCII code. All ESC sequences and message text passed between a local workstation or user program must be in EBCDIC.

- When it sends a message to the host, the IBM 3277 terminal sends all modified fields, beginning at the home position, regardless of the current cursor position. The message, however, contains the final cursor position.

  The local workstation begins transmission at the SOE (or home) and doesn't provide the cursor position. When it sends a message to the host, the 3270 Emulator usually generates the last screen location as the current cursor position.

  To ensure that all local workstation modified data between the home position and the cursor is sent to the IBM host, never key in SOE characters and always place the cursor after the last modified field.

  The IBM host must never send any SOE characters to the local workstation.

- The IBM 3270 uses the NUL character as an erase character and the SYNCH character for time fill. The UTS uses both the NUL and SYNCH characters for time fill and uses the space (SP) character as an erase character.

  When it receives messages from the IBM host, the 3270 Emulator converts NUL characters to UTS spaces. When it sends messages to the IBM host, the 3270 Emulator suppresses all NUL characters, but includes local workstation trailing spaces.

- While the IBM 3270 acknowledges message receipt to the IBM host after the message is successfully transferred to the IBM display terminal, the 3270 Emulator acknowledges message receipt after the message is queued in a network buffer. Thus, when a program or local workstation terminates or aborts a session after the 3270 Emulator receives, but before it delivers, a message bound for that end-user:

  - the message is thrown away; but
  - the IBM host is not aware of the loss.

- When the local workstation enters system mode, messages bound for that workstation from the IBM host aren't delivered until workstation mode is resumed. These messages tie up network buffers until they're delivered and, if the workstation closes the session ($$SOFF) before resuming workstation mode, the messages are lost.

- Don't initialize a 3270 switched line until the first session attaches to a terminal on that line.

3.3.2.1.9.3. Console and Terminal Operator Interfaces

With the 3270 Emulator, there are no changes to the way you interface with the console, but there are some changes to the way you interface with terminals.

To open an ICAM session, the workstation must be in system mode. You establish a global session through the standard sign-on ($$SON xxxxyyyy), which links the local workstation to the *pseudo* IBM terminal. In the sign-on, you identify yourself (xxxx) and the other end-user (yyyy). For example, you may specify $$SON LWS3TRM2. If the pseudo IBM terminal is in use or is unavailable, the sign-on is rejected. If the session can be initiated, ICAM displays the messages:

```
SPERRY UNIVAC DCA NETWORK, LEVEL xx, NODE yyyy
SESSION PATH OPEN
```

Once the session is opened, the workstation is in data mode. At that point, you send the cursor to HOME and clear the screen. You then can send messages to and receive messages from the IBM host.

You close the ICAM session by performing the standard sign-off ($$SOFF) and then logging off (LOGOFF).

Note that all generation requirements, procedures, guidelines, and restrictions dictated for ICAM local workstations apply here.

3.3.2.1.9.4. System Generation

The 3270 Emulator requires new parameters for network generation. You must use:

- a LINE macroinstruction to define the line to the IBM host; and
- one or more TERM macroinstructions to define the IBM pseudo terminals on that line.

The LINE macroinstruction identifies each 3271 inverted line to the IBM host. For the DEVICE parameter, specify INV3270. Because the 3270 Emulator interprets all data in EBCDIC, specify NO for the XLATE parameter of the LINE macroinstruction. Default values for other parameters are:

- Input=4 and output=6 for the RETRY parameter.
- input=5 minutes and output=3 for the TIMEOUT parameter. Do not specify TIMEOUT for the 3270 Emulator.

The TERM macroinstruction identifies the device as a 3277 display station. The terminals online to the IBM host are pseudo terminals in the OS/3 ICAM environment; they do not physically exist, except when they're linked to a local workstation in a dynamic session. To the IBM host, these terminals seem like 3277 display terminals.

You may generate the network with enough TERM macroinstructions to satisfy up to the maximum of 32 simultaneous 3270 sessions. The PASS, STRIP, and CONV parameters, discussed in 3.3.2.1.9.5.4, apply to the 3277 TERM macroinstruction. For the ADDR parameter of the TERM macroinstruction, you must specify the control unit and device addresses. (All control unit addresses must be the same for all of the TERM macroinstructions associated with one line.) When configuring ICAM with local workstations, you must specify LWSs in the TERM macroinstruction.

The LINE and TERM macroinstructions are described in the ICAM network definition and operations user guide, UP-8947 (current version).

A program in a global environment must follow global network standards. In a global network, the paired end-users are the program and the IBM pseudo terminal.

To establish a global dynamic session, your program must use the SESCON macroinstruction and a datagram to OPEN the session with the IBM pseudo terminal. Note that messages generated for display on the called IBM terminal are not sent to the IBM host.

To establish a static session, generated by the SESSION macroinstruction, your program doesn't need the dynamic sessioning code, but, in the network generation, the 3277 TERM macroinstruction must specify INPUT=LOCAP name.

3.3.2.1.9.5. Message Formatting

When you write a program to communicate with the IBM host, your program must follow strict message format rules because it takes the place of the 3277 display terminal operator and the 3271 controller message formatting logic. Other considerations are:

- IBM commands and orders;
- the PASS, CONV, STRIP, and DICE options;
- screen formatting; and
- the IBM application it interfaces, so it can correctly interpret incoming messages and generate responses.

3.3.2.1.9.5.1. IBM Commands

Table 3-17 describes the IBM commands supported by the 3270 Emulator. The commands not supported are noted.

Table 3-17. IBM Commands

Command	EBCDIC Hex	EBCDIC Graph	Remarks
WRITE	F1	1	
ERASE/WRITE	F5	5	
DIAGNOSTIC WRITE	F9	9	Diagnostic commands are not supported. Use System 80 diagnostic facilities to validate System 80/local workstation hardware.
DIAGNOSTIC READ	7A	:	
READ BUFFER	F2	2	Not supported
READ MODIFIED	F6	6	
COPY	F7	7	Not supported.
ERASE ALL UNPROTECTED	6F	?	

- Write Command

Write command transfers display data from the IBM host to the local workstation or program. The write data stream to the 3270 Emulator consists of binary synchronous communication procedure framing characters (STX, ESC, ETX), the write command code, a write control character (WCC), orders, and data. Figure 3-4 shows how a message is received from the IBM host.

STX	ESC	WRITE	WRITE CONTROL CHARACTER	SET BUFFER ADDRESS ORDER	ORDERS/ DATA	SET BUFFER ADDRESS ORDER	ORDERS/ DATA	ETX

Figure 3-4. Sample Format of a Message from the IBM Host

If an ETX follows a write command without including a write control character, the 3270 Emulator handles the message as if there were an all-zero write control character. It interprets the order/data byte immediately following the command code as a write control character.

The following diagram shows the structure of the write control character byte and Table 3-18 shows how it is translated into a UTS command sequence.

X	X	printer format	start printer	sound alarm	restore keyboard	reset modified data tag
0	1	2 3	4	5	6	7

Table 3-18. Write Control Character (WCC) Format

Bit Number	Remarks	
	3271/Emulator	UTS/Emulator
0, 1	Determined by the contents of bits 2 through 7	
2, 3	Printer format	Not supported
4	Start printer	Not supported
5	Sound alarm	Not supported
6	Restore keyboard	If zero, keyboard is kept locked (i.e., a DC4 is inserted at the end of text).
7	Reset modified data tag	This bit will be converted to an 'ESC u' (clear changed bit).

- Erase/Write Command

The erase/write command first erases the entire screen, and then transfers data from the IBM host to the workstation or program. On the IBM 3270 control unit, the device buffer, attribute bytes, and protected fields are cleared, and the cursor is positioned to character location 0. If no write control character is sent, the IBM 3270 doesn't erase the buffer.

The erase/write command causes the following UTS screen control sequences to precede data sent to a UTS:

Cursor-to-home code ESC e
Erase display ESC M

If no write control character is included (an ETX follows the erase/write command), the 3270 Emulator positions the cursor to HOME, but it doesn't erase the buffer.

- Read Modified Command

The read modified command transfers data from the workstation or program to the IBM host. A local workstation sends only modified fields to the host. The 3270 Emulator transfers a long message to the host if a text message is queued for output; it transfers a short message if you press a program access (PA) key.

The host sends the 3270 Emulator a read modified command only when:

- Transmission to the host computer is interrupted. Transmission is stopped and the 3270 Emulator sends an EOT.
- Transmission to the host computer is already suspended. Before receiving another poll, the 3270 Emulator retransmits the entire message. If no read modified command is received, interrupted messages are thrown away.

If it receives a read modified command at any other time, the 3270 Emulator transfers a short text message back to the host, indicating 'no key depressed'.

Figures 3-5 and 3-6 show sample formats for long and short messages transmitted by the 3270 Emulator.

STX	CONTROL UNIT ADDRESS	DEVICE ADDRESS	AID X'7D' *	CURSOR ADDRESS **	SET BUFFER ADDRESS SEQUENCE	DATA	SET BUFFER ADDRESS SEQUENCE	DATA	ETX

* AID (attention identifier)-X'7D' (ENTER key pressed) byte or program function key pressed, generated by 3270 Emulator.

** CURSOR ADDRESS = last screen position

Figure 3-5. Sample Long Message Format for Read Modified Command

STX	CONTROL UNIT ADDRESS	DEVICE ADDRESS	AID X'60' *	X'40'	X'40'	ETX

* AID (attention identifier)-X'60' (no key pressed) or program attention key pressed, generated by 3270 Emulator.

Figure 3-6. Sample Short Message Format for Read Modified Command

- Erase All Unprotected Command

The erase all unprotected command:

- erases all unprotected fields on the workstation display from beginning of the screen;
- resets all MDT bits (equal to the CHANGED bit in a UTS field control character); and
- positions the cursor to the first character of the first unprotected field, or to HOME if it's an unformatted screen.

Figure 3-7 shows the message format for the erase all unprotected command.

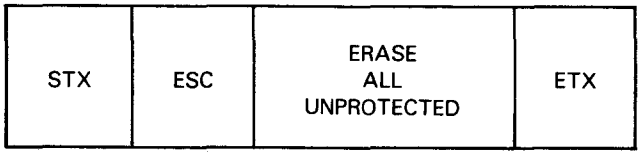


Figure 3-7. Sample Message Format for Erase All Unprotected Command

The erase all unprotected command is translated into the following UTS screen control command sequence:

Cursor-to-home code	ESC e
Clear changed	ESC u
Erase unprotected data	ESC a
Forward tab	HT

3.3.2.1.9.5.2. IBM Orders

Orders are buffer control sequences embedded in the data to position, define, and format data for the 3270 buffer and screen. They can be from 1 to 4 bytes long. Table 3-19 describes the IBM orders supported when you support a local workstation or when you specify the CONV parameter of the TERM macroinstruction.

Table 3-19. Supported IBM Orders

Order	Abbreviation	Order Code Byte 1	Byte 2	Byte 3	Byte 4
Start Field	SF	X'1D'	Attribute byte		
Set Buffer Address	SBA	X'11'	1st address byte	2nd address byte	
Insert Cursor	IC	X'13'			
Program Tab	PT	X'05'			
Repeat to Address	RA	X'3C'	1st address byte	2nd address byte	Repeated Character
Erase Unprotected to Address	EUA	X'12'	1st address byte	2nd address byte	

- Start Field Order

This order signals the IBM 3270 control unit that the next byte in the data stream is an attribute byte (ATB). The attribute byte occupies the space for one screen character on the IBM 3277 display device.

When the 3270 Emulator detects a start field order, it treats the next byte as an attribute byte and, if the buffer address is known, translates it into a UTS long field control character (FCC) sequence. If the current buffer address is not known, the start field/attribute byte is replaced by a space character. Thus, the resulting UTS screen is not field formatted correctly, but the screen image is identical to the 3277 display.

Table 3-20 shows how attribute bytes are translated to UTS field control characters.

Table 3-20. IBM Attribute Byte Translation to Field Control Character (FCC)

IBM 3277 Attribute Byte	UTS Field Control Character	Remarks
Protected	Protected (auto skip)	
Unprotected	Unprotected	TAB bit always set in field control character
Alphanumeric	Any input	
Numeric	Numeric	
Display	Normal intensity	
Intensified display	Low intensity	
Nondisplay		Not supported, field is displayed
Auto skip		Not supported, but a protected field acts like it
Modified	Changed	
Light pen detectable		Not supported

NOTE:

To support the program tab order, the 3270 Emulator always sets all unprotected fields (field control character) with a TAB bit.

- Set Buffer Address Order

  The set buffer address order (SBA) specifies the starting or continuing address for a write operation on an IBM 3270 control unit. The SBA is connected to a UTS cursor positioning sequence (ESC VT Y X SI). In a user program, if DICE=ON and CONV MODE is set, the SBA sequence is connected to a DICE sequence.

  When no set buffer address order follows the write control character in a write data stream to the 3270 Emulator, the data appears on the UTS screen after the current cursor position.

  If a start field order follows the set buffer address order, the set buffer address/start field orders are converted to a UTS field control character sequence. The set buffer address characters are converted to a row-column address sequence (US Y X ...), and the start field order is converted to the m and n bytes of the UTS field control character sequence. This results in a long field control character (FCC) sequence (US Y X M N).

- Insert Cursor Order

  This order repositions the cursor to the location specified by the current buffer address location on an IBM 3277 display device. If an insert cursor order is specified in the IBM message, the 3270 Emulator puts a VT sequence at the end of the message to position the cursor. The current buffer address is converted to a cursor address sequence for the UTS terminal:

  ```
  ESC VT Y X SI
  ```

  where Y is the row and X is the column on a UTS screen.

  In a user program, if DICE=ON and CONV MODE is set, a DICE sequence is appended to the end of the message.

  A set buffer address order should precede the insert cursor order.

  If the current buffer address is not known, the insert cursor order is stripped, and the UTS cursor is positioned at the end of the message.

- Program Tab Order

  The program tab order positions the buffer address to the next unprotected field. When formatting a UTS screen, the 3270 Emulator marks all unprotected fields with a TAB bit in the field control character byte.

  When a program tab order follows:

  - A running text in a write data stream from the IBM host, an erase-to-end-of-field (ESC K) and a forward-tab code (HT) are inserted in the message to a UTS terminal.

  - A write control character order, only a forward-tab code (HT) is inserted in the message.

- Repeat to Address Order

  The repeat to address order stores a character from the current buffer address up to, but not including, the stop address on an IBM 3270 control unit. Attribute bytes are overwritten. No equivalent order exists on the UTS.

  If the current buffer address is known, the 3270 Emulator inserts the repeat character ((stop address) – (current buffer address)) times in the data stream to the UTS.

  If the current buffer address is not known, the repeat character is not sent to the UTS, but the UTS cursor is repositioned.

3.3.2.1.9.5.3. Attention Identifier (AID) Byte

The attention identifier byte begins the text of each message sent to the IBM host. It identifies the key you pressed to initiate the transfer to the host. The IBM 3270 sends:

- A short read message when you press a program access (PA) key.
- A long read message when you press a program function (PF) key or ENTER key.

The 3270 Emulator sends:

- A short read message (attention identifier byte only) when simulating the program access key.
- A long read message (AID (attention identifier) = 7D) when simulating a program function key (F5–F22) or when you press the ENTER (XMIT) key.

The 3270 Emulator uses the 22 local workstation function keys (F1–F22) to simulate the IBM 3270 program access (PA1–PA4) and program function (PF1–PF18) keys. IBM program function keys 19 to 24 are not supported on the local workstation. To simulate the IBM keys, your user program must set the TM#DSPEC auxiliary function field in the output DTFCP (see 2.21.1), and can specify up to 24 function keys.

Table 3–21 shows attention identifier bytes supported on local workstations and in user programs.

Table 3-21. Attention Identifier (AID) Byte Support

IBM AID	AID HEX	GRAPHIC	USER PROGRAMS-TM#DSPEC	LOCAL WORKSTATIONS
NO AID (DISPLAY)	60	No graphic		READ MODIFIED COMMAND
NO AID (DISPLAY) PRINTER	E8	Y		Not supported
ENTER KEY	7D	'	TM#PENTR	XMIT KEY
SELECTOR LIGHT PEN	7E	=	TM#PLPEN	Not supported
OPERATOR IDENTIFIED CARD READER	E6	W	TM#PMGRN	Not supported
PA1 KEY	6C		TM#PFRMW	LWS F1 KEY
PA2 (CNCL)	6E	>	TM#PAK2	LWS F2 KEY
PA3 KEY	6B	,	TM#PAK3	LWS F3 KEY
PA4 (CLEAR)	6D	-	TM#PCLER	LWS F4 KEY
PF1 KEY	F1	1	TM#PFK5	LWS F5 KEY
thru	. . .	. . .	. . .	. . .
PF9 KEY	F9	9	TM#PFK13	LWS F13 KEY
PF10 KEY	7A	:	TM#PFK14	LWS F14 KEY
PF11 KEY	7B	#	TM#PFK15	LWS F15 KEY
PF12 KEY	7C	@	TM#PFK16	LWS F16 KEY
PF13 KEY	C1	A	TM#PFK17	LWS F17 KEY
thru	. . .	. . .	. . .	. . .
PF18 KEY	C6	F	TM#PFK22	LWS F22 KEY
PF19 KEY	C7	G	TM#PFK23	Not supported
thru	. . .	. . .	. . .	. . .
PF24 KEY	4C	<	TM#PFK28	Not supported
TEST REQ KEY	F0	0	TM#PTREQ	Not supported

These restrictions apply:

1. The buffer address must be less than the stop address.

2. No screen wrapping is supported.

3. The UTS field format is not changed. The UTS doesn't reset the field control character.

- Erase Unprotected to Address Order

  This order erases all unprotected fields from:

  - the current buffer address, or

  - an address given in a set buffer address order on an IBM 3277 display terminal,

  to the end of the display on a UTS (on the IBM 3270, it erases all unprotected fields to the address specified).

  No equivalent order exists on the UTS. The erase unprotected to address order causes the following sequence:

  Erase unprotected data ESC a
  Cursor positioning sequence ESC VT Y X SI

3.3.2.1.9.5.4. Program Input

The input message format sent by the host to System 80 conforms to the message format described in Figure 3-4. The 3270 Emulator: strips the communications envelope (STX....ETX); and passes the rest of the message, consisting of commands and text, to the program in a format specified by the PASS, CONV, STRIP, and DICE parameters of the TERM macro. See the ICAM network definition and operations user guide, UP-8947 (current version) for details on the PASS, CONV, DICE, and STRIP parameters of the TERM macroinstruction.

- PASS Parameter

  When you specify the PASS parameter, the message text that System 80 receives is passed unmodified into your program's work area. The text contains IBM 3270 commands, IBM orders, and text data as follows:

ESC	WRITE or ERASE/WRITE COMMAND	WRITE CONTROL CHARACTER	SET BUFFER ADDRESS a1 a2	Data 1	SET BUFFER ADDRESS a1 a2	Data 2

To respond to the host message, your program must interpret the IBM command and orders along with the data. This interpretation may require emulating a 3277 screen image within your program. The DICE specification is ignored.

- CONV Parameter

  When you specify the CONV parameter, the 3270 Emulator converts the message received from the host into workstation format. IBM commands and orders are converted to local workstation ESC sequences. To respond to the host message, your program must interpret the DICE and local workstation sequences along with the data.

  If DICE=ON, all set buffer address orders are converted to DICE=SET COORDINATE, except when a start field order directly follows a set buffer address, resulting in a long field control character sequence. All other orders are converted to the workstation ESC.

- STRIP Parameter

  When you specify the STRIP parameter, the message text received from the host is stripped of all IBM orders and the command sequence. The DICE specification is ignored – nothing is converted.

  If the message received from the host is:

STX	ESC	WRITE or ERASE/WRITE	WRITE CONTROL CHARACTER	SET BUFFER ADDRESS a1 a2	text	INSERT CURSOR

the 3270 Emulator strips the orders and command (in the example, the set buffer address and insert cursor orders and the write or erase/write command) and places the text into the work area.

3.3.2.1.9.5.5. Program Output

The output message format sent to the host must conform to the long or short message format described in Figures 3-5 and 3-6. The 3270 Emulator provides the communications envelope (STX, CONTROL UNIT ADDRESS, DEVICE ADDRESS, ... ETX/ETB). Your program must provide the remainder of the message content (ATTENTION IDENTIFIER, CURSOR ADDRESS, data, and SET BUFFER ADDRESS).

Your program specifies the attention identifier (AID) byte for all output messages by setting TM#DSPEC in the DTFCP with the attention identifier value.

If you specify:

- an IBM program attention (PA) key, the work area prefix must be zero to indicate that a message is not being sent; or

- an IBM program function (PF) key or ENTER key, the work area prefix must specify the message length and the work area must contain the message. You specify the ENTER key with an AID (attention identifier) = TM#PENTR.

You must set up the outgoing message in the work area specified by the PUTCP. The message format depends on the PASS, CONV, STRIP, or DICE parameter you choose. See the current version of ICAM network definition and operations user guide, UP-8947.

- PASS parameter

  When you specify the PASS parameter, the message is sent to the host without modification, and the program must generate all messages in IBM format. The first two characters make up the cursor address; the rest of the message contains the set buffer address and the data. The DICE specification is ignored, and no DICE conversion is performed.

- CONV parameter - DICE=ON/OFF

  When you specify the CONV parameter, messages are converted depending on the DICE specification and whether the 3277 screen is formatted or unformatted.

  A formatted screen is laid out by the IBM host software with fields and the associated attributes. In a message from an unformatted screen, the IBM host doesn't expect to receive any set buffer address. Your program must generate messages in local workstation or DICE format. The 3270 Emulator then converts the message into the IBM format and sends it to the host.

  In the following cases, the 3270 Emulator generates a cursor address of row=24 column=80 (shown as CURSOR ADDRESS = 24/80):

  1. For formatted screens and DICE=OFF, when the program generates a message made up of a sequence of modified fields:

UNIT SEPARATOR CHARACTER y x m n	FIELD 1	UNIT SEPARATOR CHARACTER y x m n	FIELD 2

the 3270 Emulator sends the message:

CURSOR ADDRESS 24/80	SET BUFFER ADDRESS a1 a2	FIELD 1	SET BUFFER ADDRESS a1 a2	FIELD 2

2. For formatted screens and DICE=ON,

   - when the program generates a message:

DICE (y,x)	FIELD 1	DICE (y, x)	FIELD 2

the 3270 Emulator sends the message:

CURSOR ADDRESS 24/80	SET BUFFER ADDRESS a1 a2	FIELD 1	SET BUFFER ADDRESS a1 a2	FIELD 2

- when the program generates a message:

DICE (y,x)	DICE (y,x)	FIELD 1	DICE (y,x)	FIELD 2

the 3270 Emulator converts the first DICE sequence to the CURSOR ADDRESS and sends the message:

CURSOR ADDRESS y/x	SET BUFFER ADDRESS a1 a2	FIELD 1	SET BUFFER ADDRESS a1 a2	FIELD 2 ...

3. For unformatted screens and DICE=OFF, when the program generates a message:

text ...

the 3270 Emulator sends the message:

CURSOR ADDRESS 24/80	text ...

4. For unformatted screens and DICE=ON, when the program generates the message:

DICE (y,x)	text ...

the 3270 Emulator converts the first DICE sequence to the CURSOR ADDRESS and sends the message:

CURSOR ADDRESS y/x	text ...

NOTE:

If DICE=ON, and the message from the IBM host contains an INSERT CURSOR (IC) order, the 3270 emulator will strip the IC order from the message and append a DICE sequence (reflecting the IBM buffer address when the IC was encountered) at the end of the message.

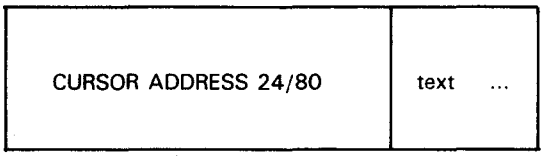
- STRIP parameter

When you specify the STRIP parameter, the message sent to the IBM host should only be text. A user program must be unformatted (no set buffer address) in its work area.

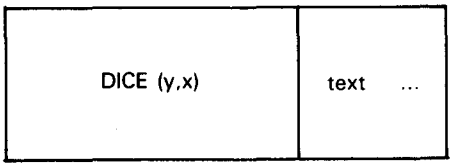
If your program requires a CURSOR ADDRESS, it must specify DICE=ON and include the DICE sequence as the first part of the message.

For DICE=OFF, when the program generates a message:

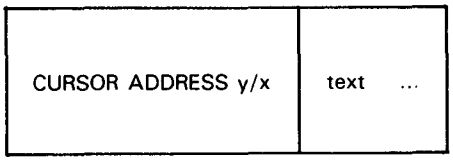
the 3270 Emulator sends the message:



For DICE=ON, when the program generates the message:



the 3270 Emulator converts the first DICE sequence to the CURSOR ADDRESS and sends the message:



3.3.2.1.9.5.6. DICE Support

The DICE function is supported when a user program is in session with an IBM host. Table 3-22 shows how and when the DICE sequences are converted to IBM set buffer address sequences, depending on whether you specify the CONV, STRIP, or PASS parameter of the TERM macroinstruction.

Table 3-22. Converting DICE Functions

Parameter Specified in TERM macro	INPUT (IBM host to program)	OUTPUT (3270 Emulator to IBM host)
CONV	Program - only IBM set buffer address a1 a2 orders converted to set coordinate DICE function. Local workstation - DICE specification ignored.	Program with DICE=ON or local workstation - 3270 Emulator converts DICE function (X '1001 m_a n_a ') to IBM set buffer address sequence. Program with DICE=ON and beginning with double DICE sequences - the first DICE sequence is converted to a1 a2 coordinates (specifying current cursor address to host) and placed after attention identifier byte.
STRIP	DICE specification ignored; text not modified.	Text not modified. If DICE=ON, DICE sequence at beginning of message converted to cursor address.
PASS	DICE specification ignored; text not modified.	

3.3.2.1.9.5.7. Data Code Considerations

Some EBCDIC data codes and graphics used by the OS/3 system and the local workstation don't exist or are different from those on the IBM 3277.

The first three entries in Table 3-23 show EBCDIC codes causing different display graphics on the IBM 3277. The other seven entries show OS/3 local workstation codes and graphics causing nondisplays on the IBM 3277 display terminal.

Table 3-23. Comparison of Data Codes and Graphics

OS/3 EBCDIC (Hexadecimal)	OS/3 Local Workstation Graphic	IBM 3277 Graphic
4A	[OPEN BRACKET	CENT SIGN
4F	] CLOSED BRACKET	| LOGICAL OR
5F	^ CIRCUMFLEX	LOGICAL NOT
07	DELETE	NON-GRAPHIC
6A	| VERTICAL LINE	
79	' GRAVE ACCENT	
A1	OVERLINE	
C0	{ OPEN BRACE	
D0	} CLOSED BRACE	
E0	\ REVERSE SLASH	

IBM printer control new-line (NL), end-of-medium (EM) and form feed (FF) characters, displayed as 5, 9 and < on the IBM 3277, are not displayed on the local workstation.

3.3.2.1.9.5.8. Status and Sense Bytes

The 3270 Emulator sets up status and sense (S/S) bytes to inform the host of how some Emulator and UTS conditions conform to IBM 3270 conventions. A status message is returned to the IBM host in response to a general or specific poll. Figure 3-8 shows the status message format.

SOH	%	R	STX	CONTROL UNIT POLL ADDRESS	DEVICE ADDRESS	S/S 0 (1ST STATUS BYTE)	S/S 1 (2ND STATUS BYTE)	ETX

Figure 3-8. Sample Format for Status Message

The 3270 Emulator generates S/S bytes when:

1. A session is made active to a 3270 Emulator terminal.
2. Either no session is active or a session is aborted.
3. It receives an unsupported command.
4. It receives a read modified command after a specific poll.
5. The 3270 Emulator receives an illegal buffer address or incomplete order sequence on a write or erase/write command.
6. It receives an invalid command sequence.

3.3.2.1.10. UTS 400/UTS 4000 Terminal Capability Comparison

Table 3-24 summarizes the capabilities that are present for the various types of UTS 400/UTS 4000 terminals. The lack of a capability may be due to a hardware, firmware, or software consideration.

Table 3-24. Capability Comparison of UTS 400/UTS 4000 Terminals (Part 1 of 2)

Capability	UTS 400 Native Mode	UTS 400 UNISCOPE Mode	UTS 400 Text Editor	UTS 4000
Automatic disconnect	Yes ①	No	No	Yes
Function keys (F1-F4) Function keys (F5-F22)	Yes Yes	Yes No ②	No No	Yes Yes
Automatic retry - auxiliary devices	Yes	No ③	Yes	Yes
Shared peripheral operation	Yes	No ④	No	Yes
Operator-initiated power on confidence test (POC)	Yes	Yes ⑤	Yes	Yes
Screen bypass	Yes	Yes	No	Yes
User programmability	Yes	No	No	Yes ⑦
Katakana capability	Yes	No	No	Yes
True mode strapping	No	No	Yes ⑥	No

Table 3-24. Capability Comparison of UTS 400/UTS 4000 Terminals (Part 2 of 2)

NOTES:

① Special feature required

② Although available, they are invalid input to the ICAM RDH. To prevent inadvertent usage, the FCC/PROTECT switch must be set to PROTECT.

③ Do not strap terminal with this capability.

④ When defined as a UNISCOPE terminal, the program must ensure that contention for an auxiliary device does not occur. Contention can occur between the terminal operator and the program, or the program itself can cause contention. Operational procedures can prevent the former, while the latter can be prevented by the program in addressing all auxiliary output to one station.

⑤ Should not be used while transmission between host and terminal is in progress. The DLE6 response (successful POC test) must not be transmitted to the host. It must be disabled.

⑥ Software operation requires that the true mode must not be strapped.

⑦ Not available on the UTS 20 single station.

3.3.2.2. DCT 500 Series/TELETYPE Terminals

The DCT 500/DCT 524/DCT 475/TELETYPE/UTS 10 remote terminals are handled with the DCT 500/TTY RDH. The remainder of this section discusses the following aspects of the appropriate terminals:

- Considerations common to all supported terminals
- TELETYPE mode support
- Semiautomatic mode support
- Tape cassette operations (DCT 524)

3.3.2.2.1. Common Considerations

The areas of support common to DCT 500, DCT 524, DCT 475, UTS 10, and TELETYPE terminals are:

- Character Parity

  Assume even-character parity.

- End of Message

  ETX or control C is the end of message key and must be used at the end of each message for non-Baudot terminals. FIGS H LTRS is the end of message sequence for Baudot TTY.

- Cancel/Erase Characters

  To cancel a character (backspace), the control H key is used for non-Baudot terminals. To cancel an entire message, control X is used followed by control C (ETX) for non-Baudot.

- Input Time-out

  If an input message is entered at a rate of less than 80 characters (line buffer size) within 512 seconds, the entire message is discarded as though it were canceled. Note that if input DICE is used, the backspace should not be used immediately after a form feed, carriage return, or line feed, since the results vary depending upon the next input character. Use the cancel message function in these cases.

- Break

  If the BREAK key is held down while the terminal is receiving a message from the RDH, the message is terminated. After the terminal again sends input, the terminated message is retransmitted. If the BREAK key is held down while the terminal is sending input to the RDH, the input message is discarded as though it were canceled.

- Preparing Paper Tape Input Manually

  - At least several time fill characters (DELETE or RUBOUT) should precede the text message on tape.

  - If tape contains more than one message, each ETX must be followed by an X-OFF/DC3 with at least several time fill characters (DELETE/RUBOUT).

  - Reading tape remotely - When the RDH receives a read request, it builds the necessary control/turn-on message and sends same to the tape reader. Therefore, the tape must be loaded and the TD-CALL-TN switch on. If the terminal is unattended and will be connected later, nothing more need be done than ensure that all switch settings are on. However, if the situation is more dynamic, requiring some terminal-user program communication, then the operator must ensure that the tape is loaded and ready before the user program is notified that the tape reader is ready. This can be done by keying in the user-program/operator protocol message and then moving the tape to the load point (X-OFF/DC3) by hitting the TD-ON/PROCEED key before the ETX is sent. If the tape was prepared offline without an X-OFF/DC3 at the beginning, the TD-ON/PROCEED step is not necessary; otherwise, it is essential. With notification of the tape device being ready, the user program issues a read and the tape is read.

    NOTE:

    The RDH puts all necessary control characters/sequences into the commands sent to the tape device. The user program does not need to insert any control characters into the message.

- Terminal Marked Down

  When ICAM marks a terminal down as a result of an error, output to that terminal is held by ICAM until input comes in from that terminal, causing it to be marked up again, or until the terminal is marked up from the console.

- Restrictions

  - The special functions of auxiliary device write and auxiliary device read are assumed when the auxiliary device field indicates a tape punch or a tape read, respectively (Table 3–7).

  - If the auxiliary device index field is zero, the primary device printer or keyboard is assumed according to whether the function is output or input, respectively.

  - A terminal that is strapped to do an automatic line feed with a carriage return (or vice versa) cannot utilize the complete capability of DICE. Care should be used in employing such terminals in message switching networks.

  - TTYs/DCT 475/UTS 10 terminals and DCT 500/DCT 524 terminals in TTY mode cannot be multidropped (more than one terminal per line).

3.3.2.2.2. TELETYPE/DCT 475/UTS 10 and TTY-Mode DCT 500/DCT 524 Terminal Considerations

3.3.2.2.2.1. Reference Documents

Reference documents for particular teletypewriters can be obtained from the Teletype Corporation. See 3.3.2.2.3 for SPERRY UNIVAC terminal reference documentation.

3.3.2.2.2.2. Operational Considerations

- Preparing Paper Tapes Manually

  If the tape contains a single message, the ETX may be punched on the tape following the text message or it may be entered from the keyboard after the tape has been read. Note that if ETX is on the tape, and no X-OFF/DC3 follows the ETX, there is the possibility of garbled output messages sent to the TTY while the tape reader is reading the blank end of the tape following the ETX.

- Reading Paper Tape Manually

  Tape input can be sent manually by the terminal operator, without any remote commands, by moving the tape to the load point (beyond the X-OFF/DC3 if present) and then pressing the TD-ON/PROCEED key. Input received in this fashion is treated as though it were from the keyboard, since RDH did not receive a read command from the user program.

- Reading TTY-Mode DCT 500 Paper Tape

  - A paper tape with no DEL characters following the DC3 character will not be read correctly by a TTY-mode DCT 500 on a half-duplex line at speeds below 300 baud without a modified feature board. This board can be obtained through Sperry Univac personnel by requesting Communications Interface Feature Board 2818420-00 which provides for turnaround delay.

  - This kind of tape without DEL characters following a DC3 character can be produced when the user program directs output to the tape punch since the tape punch will strip out the necessary DEL characters inserted by the RDH. This kind of tape could also be produced offline if a terminal operator did not insert DEL characters following the DC3.

- Output to Paper Tape

  - For the special punch unit available on a TTY (i.e., the TTY rotor unit), or a tape punch device which responds to DC2 and DC4 characters, no operator intervention is required for punching paper tape.

  - For the standard punch units on a TTY, and punch or write units on DCT 500/DCT 524, correct timing of turning on the paper tape punch, or write to a magnetic tape, is essential and is the responsibility of the TTY terminal operator. Both keyboard and punch switches are located on the keyboard. If the punch/write switch is turned on, messages sent by the keyboard, as well as messages received, will be punched on paper tape or written on magnetic tape.

- Print Wheel

  The DCT 500/475/524 can be strapped so that if no input or output takes place within a given period of time, the print wheel stops revolving. However, some characters at the beginning of messages sent to a terminal in this condition will be lost. This problem can be resolved in one of two ways:

  - The length of time before an inactive print wheel turns off is a strappable variable of from 45 to 360 seconds. This can be overridden so that the print wheel is always moving whenever the terminal is on.

  - The user program can insert time fill characters (NULL or DEL characters) at the beginning of such intermittent output messages.

3.3.2.2.2.3. TTY-Mode DCT 500/DCT 475 Keyboard

The DCT 500/DCT 475 keyboard is selected only when the CLEAR-TO-SEND indicator is on. If not on, this light comes on when the PROCEED key is pressed.

The PROCEED key also extinguishes the INTERRUPT indicator.

3.3.2.2.2.4. TTY-Mode DCT 500/DCT 475 Keyboard Switch Settings

TTY mode DCT 500/DCT 475 switch settings are the same as for addressed-mode DCT 500/DCT 524 (3.3.2.2.3.4), except for MASTER instead of SLAVE.

Note the instructions on switch settings in 3.3.2.2.2.2.

3.3.2.2.2.5. TTY-Mode DCT 500/DCT 475 INTERRUPT Indicator

See 3.3.2.2.3.5.

3.3.2.2.2.6. Baudot TTY Considerations

Since the Baudot character set does not include the DC1, DC2, DC3, DC4 characters necessary for the remote control of paper tape devices by the RDH, no remote control paper tape device support is provided for Baudot TTYs. However, manual mode paper tape use is acceptable as described in 3.3.2.2.2.2.

Since the Baudot character set does not include the cancel (CAN) nor the backspace (BS) characters, those functions are not supported for Baudot TTYs. However, an input message can be canceled by employing the BREAK function key.

The Baudot end-of-message sequence must be FIGS H LTRS.

For the convenience of Baudot TTY operators, the RDH does an automatic LTRS shift after receiving an input space. This is to decrease the number of LTRS shifts needed. However, an LTRS shift after a space is still acceptable. Note that this makes a FIGS shift necessary after a space if FIGS is desired.

3.3.2.2.3. DCT 500 Series Addressed Mode Terminal Considerations

The DCT 500 series is a low-cost, unbuffered, asynchronous keyboard/printer terminal series similar in operation to a teletypewriter, providing up to 132 print positions. The DCT 500 series can replace existing teletypewriters with little or no change in the software handlers for point-to-point communications networks over voice-grade telephone toll lines or private lines.

The DCT 500 series can operate in a receive-only mode, a keyboard send/receive mode (TTY mode), or an automatic send/receive mode (addressed mode). The basic printer system (minimum equipment) can be expanded to include a keyboard and a 1-inch paper tape read/punch unit at any time. Additional optional equipment is available to allow for multistation operation.

The DCT 500 series is available in two additional configurations: the DCT 475 and the DCT 524.

- The DCT 475 is a TTY-mode terminal with no auxiliary paper tape device capability.

- The DCT 524 can be either TTY mode or addressed mode and includes a magnetic tape auxiliary device (TCS) which the DCT 500/TTY RDH only supports in the write and read modes.

The following subsections describe the general and paper tape use of an addressed mode DCT 500, and general use (not to include magnetic tape use) of the addressed mode DCT 524.

The general use of the DCT 475 and the TTY-mode DCT 500 or DCT 524, including paper tape use on the TTY-mode DCT 500, but excluding the magnetic tape use on the TTY-mode DCT 524, is described in 3.3.2.2.2.

The use of the magnetic tape on a DCT 524 in either TTY or addressed mode, as well as offline use of magnetic tape, is described in 3.3.2.2.4.

3.3.2.2.3.1. Reference Documents

Refer to the current versions of the following manuals for a more detailed description of the DCT 500:

- DCT 500 operator manual, UP-7832
- DCT 500 programmer manual, UP-7836
- DCT 500 general description, UP-7804

DCT 475 and DCT 524 use is included in the current versions of these manuals.

3.3.2.2.3.2. Operational Considerations

When using the DCT 500/DCT 524, note the following:

- DCT 500/DCT 524 Modes

  The DCT 500/DCT 524 can be either strapped as emulating a TTY (TTY mode) or strapped as an addressed or semiautomatic DCT 500/DCT 524. In the semiautomatic or addressed mode, output may be directed to the DCT 500/DCT 524 in the normal-or fast-select mode. In the normal mode of output transmission, the DCT 500/DCT 524 is status polled before the output is sent, to ensure that the output device is available and ready to receive. Note that it is not possible to get a satisfactory status poll response from an addressed mode DCT 500/524 output device on half-duplex lines operating at speeds less than 300 baud without a feature modification. This feature modification board can be obtained and installed by Sperry Univac personnel by requesting Communications Interface Feature Board 2818420-00 to provide turnaround delay. The fast-select mode selects one or more terminals as defined by a fast-select list (FSEL macroinstruction within the network definition) and sends the output without status polling. Fast-select list is not supported by ICAM RDHs.

- Establishing Connection

  When establishing connection, note the following considerations:

  - There is no terminal validation of addressed mode DCT 500/DCT 524 terminals since the correct device address (rid, sid, did) guarantees that the correct terminal sends or receives data.

  - All sending interrupts should be cleared by powering the DCT 500/DCT 524 on and off.

- Offline Manual Preparation of Addressed Mode DCT 500 Paper Tapes

  Paper tapes for addressed mode DCT 500 terminals may be prepared offline by following two rules:

  - Every text message must have the ETX character punched on the tape.

  - If there is more than one message on a tape, then each ETX character must be followed by a DC3 character plus three DEL characters.

- Online Reading of Addressed Mode DCT 500 Paper Tape

  Paper tape for addressed mode DCT 500 terminals can only be read via polls from the RDH that were initiated through user program issuance of a read command to ICAM. (The same is true of addressed mode DCT 524 magnetic tape.)

  - When the paper tape reader is on and the paper tape has been fed into the reader, the tape will be read when polled by the RDH.

  - See 3.3.2.2.3.4 for correct switch settings.

3.3.2.2.3.3. DCT 500/DCT 524 Keyboard

The DCT 500/DCT 524 keyboard must be selected before it can be used. The following steps are provided to guide the operator in using the keyboard.

1. The DCT 500/524 keyboard is selected when the DCT 500/DCT 524 CLEAR-TO-SEND indicator is lit. This light is turned on by the RDH if the terminal is in addressed mode or by the terminal operator (via the PROCEED key) if the terminal is in TTY mode.

2. After a message is keyed in, it is ended by simultaneously pressing the control character (CTL) and C keys, which causes the ETX (end transmission) character to be sent. At this time, the CLEAR TO SEND indicator is extinguished if the DCT 500/DCT 524 is in addressed mode.

3. If other addressed mode DCT 500/DCT 524 terminals are on the line, they will be selected in turn by the RDH. Once selected, a DCT 500/DCT 524 remains selected until an ETX character is sent or the time limit expires (times out). Addressed mode DCT 500/DCT 524 operators who are sharing the same line should be aware that, while they have control of the line, no other DCT 500/DCT 524 can operate. If all of the terminals on a line are marked down by ICAM, then the line is marked down and polled at a slower rate until it is marked up when any terminal sends input.

3.3.2.2.3.4. DCT 500/DCT 524 Keyboard Switch Settings

Addressed mode DCT 500 terminals under control of the RDH should have their switches set as shown in Table 3-25.

Table 3-25. DCT 500/DCT 524 Keyboard Switch Settings

Switch	Setting
MASTER/SLAVE	SLAVE
XMIT/OFF/REC MON	OFF
BAUD RATE 110 150 300	As desired
ON LINE/OFF LINE	ON LINE
KEYBOARD/OFF	KEYBOARD
PRINTER/OFF	PRINTER
PAPER TAPE READER/OFF	As desired where applicable
PAPER TAPE PUNCH/OFF	
FULL DUPLEX/HALF DUPLEX	HALF DUPLEX
UPPER/LOWER CASE	As desired

NOTE:

See 3.3.2.2.4 for DCT 524 settings.

Additional switch setting considerations are:

- Tape in the tape punch will only advance when TAPE FEED is pushed and TAPE PUNCH is OFF.

- The INTERRUPT (BREAK) key will function only when the keyboard is selected and the CLEAR TO SEND indicator is selected. This means that if an addressed-mode DCT 500/DCT 524 is receiving output and one desires to send the break signal to the RDH, one must turn the switches to OFF LINE and MASTER, press PROCEED, and then press INTERRUPT, all while the output message is being received. This is in order to postpone the message until input is sent from the DCT 500/DCT 524 to the RDH again. Another, but slightly different method of accomplishing the same result on an addressed mode DCT 500/DCT 524 would be to turn the switch to PRINTER OFF *before* receiving output.

- Paper/magnetic tape devices cannot be used manually on an addressed mode DCT 500/DCT 524; rather, they are strictly under control of the RDH as directed by the user program through write and read commands. The switches of the respective device must be on, however, in order to respond to commands from the RDH.

3.3.2.2.3.5. INTERRUPT Indicator

The red INTERRUPT indicator lights when the communications line to the DCT 500/DCT 524 is interrupted remotely or when the DCT 500/DCT 524 receives a DC3 (X-OFF). This latter case will occur when a tape is being punched by the RDH when it has received a write command for the tape.

3.3.2.2.4. DCT 524 Magnetic Tape Operation

Keyboard operation is described in 3.3.2.2.3, except that the KEYBOARD ON/OFF and the UPPERCASE switches have been combined into a single switch. The LINE-BY-LINE/STEP/CONTINUOUS switch is added to control reading to tape cassette. LINE-BY-LINE causes single blocks to be read; STEP causes single characters to be read; and CONTINUOUS causes tape to be read until an EOT is detected. When tape is to be read by the RDH, this switch should be set to CONTINUOUS.

3.3.2.2.4.1. Tape Cassette Loading

To write on a tape cassette, first insert the cassette in the write or CASS 2 head (right side); press the CASS 2, REWIND, and WRITE buttons to move the tape to load point.

To read a tape, insert the cassette in the read of CASS 1 head (left side); press the CASS 1, REWIND, and READ buttons to move the tape to load point.

3.3.2.2.4.2. Offline Use of Cassette (Both TTY and Addressed Mode)

The write head (CASS 2) is selected and the tape moved to load point when the OUTPUT switch is cycled (OFF, ON). With this setting, all keyins from the keyboard will be written on the tape if EOT is the last character of the message. Note that if a message is to be read by the RDH, ETX must precede the EOT.

The read head (CASS 1) is selected and the tape moved to the load point when the INPUT switch is cycled (OFF, ON). Tape is read when the PROCEED key is pressed, until the EOT character is detected.

When both the INPUT and OUTPUT switches are on, the tape in CASS 1 is copied onto the tape in CASS 2.

3.3.2.2.4.3. Online Use of Cassette (TTY Mode)

The write head (CASS 2) is selected and the tape moved to the load point when the OUTPUT switch is cycled (OFF, ON). With this setting, messages received by the terminal from the DCT 500/TTY RDH are written on the tape. Note that, in the half-duplex mode which is required for the RDH, all keyins from the keyboard also go on the tape with this switch setting.

The read head (CASS 1) is selected and the tape moved to the load point when the INPUT switch is cycled (OFF, ON). In this condition, the tape is read (until EOT character is read) whenever the PROCEED key is pressed, or a DC 1 (XON) is received by the terminal from the RDH.

3.3.2.2.4.4. Online Use of Cassette (Addressed or Automatic Mode)

When the OUTPUT switch is ON, the write head (CASS 2) will respond to addressed status polls and addressed output messages received from the RDH. The tape should have already been moved to the load point to begin.

When the INPUT switch is ON, the read head (CASS 1) will read the tape until an EOT character is detected, each time the RDH polls the read head. The tape should be moved to load point before being polled.

3.3.2.2.4.5. DCT 524 TCS Read Errors

As described in the DCT 500 Series programmer reference manual, UP-7836 (current version), when the DCT 524 TCS encounters a read error, a SYN character (16_{16} in ASCII or 32_{16} in EBCDIC) is inserted into the message. The TCS then skips to the next interblock gap, whereupon the TCS CHECK indicator lights. In this condition, the TCS will require two DC1 characters from the user program or two PROCEED keyins from the DCT 524 operator to read the next block. However, this block can be reread after the user program or the DCT 524 operator performs a backward-one-block, if the TCS is strapped correctly.

It should be noted that Sperry Univac software does not support the backward-one-block function on the DCT 524 TCS and the user must perform this function if he desires to have it.

When such a read error occurs, the RDH will give the user program the input message up to and including the SYN character transmitted by the DCT 524. You will be able to detect a DCT 524 tape read error value in the special function field TM#PAUX/TM#TTIAX.

At this time, the user program may want to accept this block and read the next. To do this, he must build a message consisting of two DC1 characters (DC1 DEL DEL DC1 DEL DEL) and issue this message as though it were to go to the printer with the

auxiliary device field set to zero. Note that the user must assume that an input request has been issued prior to this output to receive the next input block. After this, normal procedure is resumed.

However, the user may want to reread this block. He may do this by first discarding the error input and building an output message containing the backward-one-block command (see Appendix A of the DCT 500 programmer manual, UP-7836 (current version)). Then issue the output message again, as though it were going to the printer with the auxiliary device set to zero. After this, normal tape reading procedure is resumed.

It is important to note that if a read error occurs and the logical block plus TCS and terminal control characters (DC3, CR, LF, etc) on the tape exceed 132 characters, it can be difficult for the user program to determine where in his data buffers the erroneous block begins. This problem comes about since the TCS control characters are stripped off by the RDH while CR and LF characters are converted in 4-byte DICE expressions. Thus, if logical blocks on the tape, along with their control characters, are always kept less than 132 characters, such read error logic can be greatly simplified.

3.4. BATCH MODE TERMINALS

The following is a discussion of the remote terminal characteristics for those terminals that operate in a batch environment. Characteristics common to each batch terminal are listed, followed by a discussion of each supported batch terminal. The 90/30 system can also emulate a number of batch terminals.

3.4.1. Characteristics of Batch Terminals

The supported batch terminals are card-oriented terminals that generally support:

- Line printers
- Card readers
- Card punches

Batch terminals are normally used in remote job entry applications.

In addition to normal input and output, batch terminals may have the ability to send or receive any of several special commands that may affect the input/output processing.

In processing normal I/O, or in sending or receiving special commands, the software controlling the batch terminals makes use of the two fields defined in 3.2.

The usage of these fields is illustrated in 3.4.1.1 through 3.4.1.3.

3.4.1.1. Batch Terminal Output

Table 3-26 illustrates how RDHs and your user program control output functions.

Table 3-26. Batch Output Control Methods (Part 1 of 2)

Terminal	Output Function	Special RDH Considerations	Must User Supply Text?	Contents of Auxiliary Device Index Field	Contents of Special Function Field (TM#DSPEC)
Batch terminals in general	Normal output (printer)	The RDH will convert DICE to line and form feed control characters required by the remote terminal.	Yes	0	0
	Output to card punch	If the user attempts to send output to the card punch and the CCA has been generated to indicate no card punch available, the RDH will send the output to the printer. * For output send to the card punch, DICE will not apply except to signal the end of card image.	Yes	1	0
	Abort output	The RDH will send the current buffer and terminate output in the same manner as if it had received no ouput EOM (3.4.1.3).	No	0 or 1	0
DCT 2000	RDH must handle output text too long to be transmitted.	The RDH truncates the message at the print line length and continues normal processing.	Yes	0 or 1	0
IBM 2780, IBM 3741 or UDS 2000	Send output in transparent mode	The RDH will build a block with transparent envelope characters. Records are padded to the length of the output record specified to the TERM macro.	Yes	0 or 1	TM#DCTRN
	Interrupt the current input to transmit output	The RDH will send a reverse interrupt (RVI) to the remote terminal instead of the normal acknowledge. Depending on the type of terminal, the RVI request may or may not be honored (3.4.2.4.2).	No	0 or 1	TM#DCSRV
	RDH must handle output text too long to be transmitted	The RDH truncates the message at the print line length and continues normal processing.	Yes	0 or 1	0
	RDH processing of an RVI from another CPU		No	0 or 1	TM#DCRVI

Table 3-26. Batch Output Control Methods (Part 2 of 2)

Terminal	Output Function	Special RDH Considerations	Must User Supply Text?	Contents of Auxiliary Device Index Field	Contents of Special Function Field (TM#DSPEC)
IBM 3741	Send disconnect sequence	At termination of output file, RDH sends DLE EOT sequence and causes line to disconnect.	No	0 or 1	TM#DCBDC
	Send connect sequence	RDH sends special line connect sequence, thus allowing terminal to begin bidding for the line (3.4.2.4.3).	No	0 or 1	TM#DCBCN

* To properly identify a remote terminal that has a card punch, the appropriate keyword parameters to the TERM macroinstruction must be used in the network generation.

TERM ....., AUX1=(PCH), ...

If the AUX1 parameter is not properly set, no output can be sent to the card punch.

To identify an IBM 2780 that has a card punch, MODEL2 (printer and punch) or MODEL4 (punch only) must be supplied to the FEATURES keyword parameter to the TERM macroinstruction.

3.4.1.2. Batch Terminal Input

The user software requests input, which causes the RDH to begin polling for input. Hereafter, the control of the input process may be controlled by either the user or the operator at the remote terminal.

The remote terminal operator maintains initial control over input by operating the card reader at the terminal. However, after input has begun, the user software may choose to abort the input, or to terminate input temporarily in favor of output.

Table 3-27 illustrates how terminal operators, user programs, and RDHs control input functions.

Table 3-27. Batch Input Control Methods

Terminal	Input Function	Special RDH Considerations	Contents of Auxiliary Device Index Field	Contents of Special Function Field (TM#PSPEC)
Batch terminals in general	Normal input (card reader)	The RDH receives input from the remote terminal and inserts DICE, unless the TERM macro has specified no DICE.	0	0
	Abort input	The RDH will terminal input in the same manner as if it had read the end-of-card-deck (3.4.1.3).	0	0
1004, 9200, or 9300 subsystem	Remote terminal operator sends: READY HALT HALT, GO VOICE ABORT PRINT ABORT PUNCH OFFLINE READ	See 3.4.2.2.2 or 3.4.2.3.2.	0	 TM#PCRDY TM#PCRHT TMPCRHV TM#PCAPR TM#PCAPU TM#PCOFF TM#PCRRD
	Remote terminal hardware sends: END READ			 TM#PCRER
IBM 2780, IBM 3741 or UDS 2000	Remote terminal operator - sends transparent input with mode switch set on TSM TRSP.	The RDH sends the input but does not perform DICE processing.	0	TM#PCTRN

3.4.1.3. Abort Input/Output Functions

The abort input and abort output functions do not make use of the auxiliary device index field or the special function field. The abort function is controlled by the normal status field at the MCT/RDH interface.

During normal batch input or output, the RDH sends a normal continuation status back through the MCT interface for each card image or print image. If this normal status is changed, at the return to the RDH, the necessary steps are taken to abort the input or output.

3.4.2. Batch Terminals Supported

The following subsections discuss each batch terminal that is supported by ICAM. The terminals discussed are the DCT 2000, 1004 card processor, 9200/9300 subsystem, 1004 slave mode, IBM 2780, and UDS 2000 (IBM 2780/3741 emulation). Each discussion will include:

- Reference documents
- Operational considerations
- Special software support

3.4.2.1. DCT 2000

The DCT 2000 is a combination printer and reader/punch terminal designed to transfer large quantities of data efficiently over voice-grade facilities. This terminal can handle up to 250 blocks per minute. The DCT 2000 is also available without the combination card reader/punch for use as a printer terminal.

3.4.2.1.1. Reference Documents

Refer to the current versions of the following documents for a more detailed description of the DCT 2000:

- DCT 2000 operator reference, UP-7545
- DCT 2000 programmer reference, UP-7532
- DCT 2000 general description, UP-7511

3.4.2.1.2. Operational Considerations

When using the DCT 2000, note the following:

- Restrictions
  - DCT 2000 terminals cannot be multidropped or multiplexed on the same line.
  - When data cards are being read at the terminal, no output can be received until the last data card is read.
  - The only legal auxiliary device function is output to the card punch.
- Sending Data from the DCT 2000 – Attended Mode

The following procedure is recommended to send cards from the DCT 2000 in attended mode:

1. Place the DCT 2000 in the online, attended mode.
2. Press the reader CLEAR switch.
3. Place the card deck in the input stacker and press the CARD FEED switch. It is not necessary to have blank cards at the end of the deck.
4. Set the appropriate block size and transmit controls.
5. Press the TRANSMIT switch.
6. After all cards have been read and the TRANSMIT indicator light goes out, press the reader CLEAR and SEND EOT switches. The terminal can now receive output.

- Sending Data from the DCT 2000 – Unattended Mode

The following procedure is recommended to send cards from the DCT 2000 in the unattended mode:

1. Place terminal in attended mode.
2. Press the reader CLEAR switch.
3. Place the card deck in the input stacker and press the CARD FEED switch.
4. Place the terminal in the online, unattended mode.
5. Set the appropriate block size and transmit controls.
6. Press the TRANSMIT switch. EOT will be transmitted after last card is sent.

- Receiving Output on the DCT 2000 Printer

The following procedure is recommended for receiving output at the DCT 2000 printer:

1. Ensure that the TRANSMIT indicator light is not lit.
2. Press the printer CLEAR switch.
3. Place the DCT 2000 online.

The DCT 2000 can now receive output on the printer.

- Receiving Output on the DCT 2000 Punch

  Since the DCT 2000 uses the same card hopper for reading and punching cards, the handler must alert the DCT 2000 operator when the output is for the card punch. After the cards are punched, the handler notifies the operator that the punching operation is finished. The following procedure is used:

  1. When the output is to go to the punch, the handler prints the following message on the DCT 2000 printer:

     ```
     READY DCT 2000 FOR PUNCHING
     ```

  2. The handler sends a telephone alert signal (BEL) message to alert the operator that a change in destination is to occur. This message sounds the audible alarm.

  3. After the operator has readied the card hopper with blank cards, the operator sends a BEL message to the host. At this time, the handler starts punching cards. If punching does not begin within a reasonable period of time, the operator should send a BEL message to the host a second time.

- Special DCT 2000 Input

  The DCT 2000 operator can send one special input message, the BEL signal, to the handler. To do this, the following procedure is used:

  1. Place the RUN/STOP switch in the STOP position.

  2. Place the terminal in attended mode.

  3. Press the SIGNAL REMOTE switch.

  To reinitiate the communications activity, the operator should:

  1. Activate the GENERAL CLEAR switch if output was active.

  2. Clear the card reader and replace the last card read if input was active, and activate the CARD FEED switch.

  3. Press the SIGNAL REMOTE switch.

  4. Reset the RUN/STOP switch to RUN.

  5. Reset the ATTENDED/UNATTENDED switch to UNATTENDED.

3.4.2.1.3. DCT 2000 Software Support

The DCT 2000 RDH provides special software support for the remote terminal. It is a batch-oriented handler capable of operating in two modes:

- DCT 2000 RDH

  Allows communications between the Series 90 processing system and a DCT 2000 terminal (terminal mode).

- DCT 2000 Emulation (Slave Mode) RDH

  Allows a Series 90 system to emulate a DCT 2000 terminal, permitting communications between the system and a remote host computer.

In addition, if the terminal mode of the handler is chosen, the user has the option of including the punch capability of the DCT 2000 terminal.

The RDH expects to receive raw text data from the user to be transmitted to the terminal. When input from the terminal is active, the user can expect to receive single text images from the RDH. The RDH performs the following functions for you:

- Polling of the Terminal for Input

  When operating in emulation mode, the RDH does not poll but simply waits for the host to begin transmitting.

- Translation

  All output data is translated from EBCDIC to ASCII before transmission, and from ASCII to EBCDIC upon receipt of input text.

- Build Envelope

  Upon receipt of your output text, the proper control characters are inserted before and after the text before transmission. These envelope characters are stripped from all input text messages before they are passed to you.

- Device Selection

  When operating in terminal mode, the RDH inserts the proper character in the message envelope to select the printer or the punch, as specified by the user in the auxiliary device index field. In slave mode, this character is detected in the input text message and the user is notified whether the input is to be printed or punched.

- DICE Processing

  When operating in terminal mode, the RDH converts user-supplied DICE sequences in the output text to hardware form feed and line feed information for the terminal. The RDH always expects the DICE sequence to be the last four characters in a text image. When passing input data to the user, the RDH inserts the standard batch DICE end-of-line sequence, unless the user has selected the no-DICE option. When operating in slave mode, the RDH converts any hardware line feed and form feed characters within the input text to DICE sequences to control printing at the 90/30 system.

- Handshaking

  The RDH controls all handshaking with the terminal or host computer necessary to control input and output.

- Error Recovery

  The RDH controls all error recovery during input and output. You will not be notified of an error until the RDH has exceeded the specified retry count. At this time, a terminal down status is sent to your contingency address.

You should be aware of the following restrictions when operating with the DCT 2000 or DCT 2000 emulation RDH:

- A line buffer must be defined at network generation time; it must be large enough to contain the longest input or output text message expected, including all control characters.

- If the DCT 2000 terminal has the short-block feature, the 128-print-position feature, and/or a punch, they must be specified at network generation time.

- Each network buffer received from you during output must contain a single card or print image.

- When input is active, each network buffer supplied to the RDH must be large enough to contain the largest input text message that can be received, including the 4-character DICE sequence, unless the no-DICE option is selected.

- The user has the capability of aborting output or input at any time. It should be noted, however, that to ensure an orderly aborting of input, the user should execute a line release and a line request.

3.4.2.2. SPERRY UNIVAC 1004 Card Processor

The 1004 card processor is a powerful processing unit with arithmetic, logical, and editing capabilities. The following discussion assumes that the RMS-1 plugboard is being used.

3.4.2.2.1. Reference Documents

Refer to current versions of the following documents for a more detailed description of the 1004 card processor:

- Data line terminal type 3 manual, UP-7581
- Card processor operations manual, UP-3845
- Data line terminal type 1 manual, UP-3884

3.4.2.2.2. Operational Considerations

When using the 1004 card processor, note the following:

- Restrictions
  - The 1004 card processor cannot be multiplexed or multidropped.
  - The only legal auxiliary device function is:

    Output to the card punch.
- Establishing 1004 Card Processor Connection

  To establish connection from a 1004 card processor to the 90/30 system, use the following operating procedure:

  1. The card reader input magazine must be loaded. When the 1004 card processor is equipped with a Data Line Terminal 3 (DLT-3), the first card must be the site identification card (Table 3-28). The Data Line Terminal 1 (DLT-1) does not use the site identification card. A 1004 input deck may follow immediately behind the site-id card. If there is no 1004 input deck, five or six blank cards should follow the site-id card as padding for the card reader.
  2. The card punch, if available, must be in the ON status even though it may not be used. When the punch unit is to be used, the PCH CHECK switch/indicator must be in the ON position.
  3. The 1004 card processor operator must press the following switches:

     START
     CLEAR
     FEED
     RUN
  4. Alteration switch 1 must be set and switch 4 pressed. This causes the 1004 card processor to send the READY message at 5-second intervals until the message is acknowledged by the handler.

Table 3-28. DLT-3 Site Identification Card

Card Column	Contents	Comments
1-6	ssss△2	Constant
7-11	xxxxx	5-character site-id that must agree with the site-id specified in the 1004 TERM macro keyword answer
12	B	Constant
13-17	Blank	
18-21	SSSS	Constant
22-28	Blank	
29	B	Constant
30-31	Blank	
32-35	4321	Constant
36-37	xx	Defines the type of 1004 card processor configuration XX_{16} System Bits per Second 05 1004 1 2400 09 1004 11 2400 06 1004 1 2000 11 1004 11 2000
38-41	Blank	
42-45	SSSS	Constant
46	Blank	
47	7	Constant
48-52	Blank	
53	B	
54-62	Blank	

After the preceding steps have been performed, the 1004 card processor must be physically connected with the host processor. This physical connection is done by dialing either from the 90/30 system to the remote 1004 card processor or from the remote 1004 card processor to the 90/30 system.

After the physical connection has been made, the user controls the processing through the MCT interface. If the first user action is an input MCT, the RDH processes the site-id from the remote terminal. If the site-id does not match the site-id specified in the communications control area, an error status is sent back through the network interface. If the site-id is good, the RDH begins polling for input. If the initial user action is an output MCT, the RDH processes the output and does not process the site-id.

- Reading Cards

  In order to read cards on the 1004 card processor, use the following procedure:

  1. Ready the cards in the card reader. The card deck must terminate with an end-of-deck card (7 and 8 multiple punch in columns 1 and 2).

  2. Press the START, CLEAR, FEED, and RUN switches.

  3. Set switch 4 to OFF. Then set switch 1 to OFF.

  4. Press manual alteration switches 2 and 4 to send the cards. Cards are read until the end-of-deck card is encountered.

  If the input deck was initially loaded immediately following the site-id card, only steps 3 and 4 must be performed to read the card deck.

- Special 1004 Card Processor Input

  The 1004 card processor or 1004 operator can send special input messages to the RDH. These messages and switch settings are summarized in Table 3-29.

Table 3-29. 1004 Card Processor Special Input Messages

Command/Message	1004 Manual Alteration Switches				Sent by 1004 Operator	Sent by 1004 Hardware
	SW-1	SW-2	SW-3	SW-4		
READY	ON	OFF	OFF	ON	X	
READ	OFF	ON	OFF	ON	X	
HALT	OFF	OFF	ON	ON	X	
HALT, GO VOICE	ON	ON	OFF	ON	X	
ABORT PRINT	ON	OFF	ON	ON	X	
ABORT PUNCH	OFF	ON	ON	ON	X	
OFF LINE	ON	ON	ON	ON	X	
END READ	-	-	-	-		X

These messages are sent by the 1004 operator by using the four manual alteration switches on the 1004 card processor. Switches 1, 2, and 3 determine the message to be sent; switch 4 sends it. Therefore, when changing from one message to another, the operator should set switch 4 OFF, reset switches 1, 2, and 3 to the new message, and set switch 4 again. The messages are sent as follows:

READY message

Set the manual alteration switches in these positions:

- Switch 1: ON
- Switch 2: OFF
- Switch 3: OFF

Press switch 4; the READY message is sent.

READ message

Set the manual alteration switches in these positions:

- Switch 1: OFF
- Switch 2: ON
- Switch 3: OFF

Press switch 4 to send the READ message.

HALT message

Set the manual alteration switches in these positions:

- Switch 1: OFF
- Switch 2: OFF
- Switch 3: ON

Press switch 4; the HALT message is sent.

When the HALT message is received, the handler will retry the current I/O operation until the HALT condition is corrected. While the 1004 card processor is in the HALT condition, the operator can perform necessary duties (e.g., change paper, clear card jams). The 1004 operator can clear the halt condition by sending a READY message.

The RDH places the HALT status into the special function field, but the user need not take action on it. It is for informational purposes only, to inform the user of the reasons for delays in communication.

HALT, GO VOICE message

Set the manual alteration switches in these positions:

- Switch 1: ON
- Switch 2: ON
- Switch 3: OFF

Press switch 4; the HALT, GO VOICE message is sent.

Operation of the HALT, GO VOICE command is identical to that of the HALT command.

NOTE:

This message does not imply that the operator can hang up the modem and use it to dial the 90/30 system operator.

ABORT PRINT message

Set the manual alteration switches in these positions:

- Switch 1: ON
- Switch 2: OFF
- Switch 3: ON

Press switch 4 to send the ABORT PRINT message.

The ABORT PRINT message is sent when the 1004 operator wishes to abort the current output (print or punch).

The RDH places the abort print status into the special function field (3.4.1.3) and continues input or output normally. The user should interrogate the special function field and take appropriate action.

ABORT PUNCH message

Set the manual alteration switches in these positions:

- Switch 1: OFF
- Switch 2: ON
- Switch 3: ON

Press switch 4 to send the ABORT PUNCH message.

The ABORT PUNCH is used in the same manner as the ABORT PRINT; however, it is used to abort the current input. It signals the user that the input received up to this time is of no value and should be ignored.

OFFLINE message

Set the manual alteration switches in these positions:

- Switch 1: ON
- Switch 2: ON
- Switch 3: ON

Press switch 4; the OFFLINE message is sent.

The OFFLINE message is sent when the 1004 operator wishes to terminate communication at the completion of the current output. Once this message is sent and output completes, there can be no further communication with the 1004 card processor until a READY message is sent.

The only message sent by the remote 1004 hardware is:

```
END READ message
```

This message is sent by the 1004 card processor when it reads the end-of-deck card.

■ Special 1004 Card Processor Output

The HALT message, which stops the 1004 card processor, is the only special output message that can be sent to the remote 1004. The RDH sends the HALT message to the 1004 after receiving the ABORT INPUT indication from the MCT interface (3.4.1.3).

3.4.2.2.3. Software Support

The 1004 Card Processor RDH provides special software support for the remote terminal.

- Compression and Decompression

  The RDH performs data compression and decompression in order to save on transmission time. Compression involves taking out nonessential blank spaces on output. Decompression involves reinserting blank spaces on input. During output compression, all trailing blanks and many internal blanks are eliminated.

  As a result of input decompression, the input buffers presented to the user may not be padded with blanks.

- OUTPUT Considerations

  The output buffer presented to a remote 1004 terminal is not printed or punched by the terminal until the next output buffer is received, or until a poll for input is received. Therefore, to force the remote terminal to print the final buffer, polling must be triggered following the final buffer.

- Reporting Special Input Status to User

  The RDH receives the special input status codes from the remote terminal and converts them into software status codes for interrogation by the user. Table 3-30 shows how the software status codes are placed into the special function field.

Table 3-30. Special Function Field Input Status Settings

Status From Remote Terminal	Status Interpretation by User	Contents of Special Function Field (TM#PSPEC)
HALT	Temporary halt in communication; no action necessary	TM#PCRHT
HALT, GO VOICE		TM#PCRHV
ABORT PRINT	Abort output	TM#PCAPR
ABORT PUNCH	Abort input	TM#PCAPU
READ	Output is active and input is read at the remote terminal, but no input MCT has been posted to the RDH.	TM#PCRRD
OFFLINE	See 3.4.2.2.2.	TM#PCOFF
END READ	End of input deck	TM#PCRER

- Abort Input/Output

  Abort input/output functions are supported as described in 3.4.1.3.

- DICE Processing

  When operating in terminal mode, the RDH converts user-supplied DICE sequences in the output text to hardware form feed and line feed information for the terminal. The RDH always expects the DICE sequence to be the last four characters in a text image. When passing input data to the user, the RDH inserts the standard batch DICE end-of-line sequence, unless the user has selected the no-DICE option.

- Error Recovery

  The RDH controls all error recovery during input and output. The user will not be notified of an error until the RDH has exceeded the specified retry count. At this time, a terminal down/not available status is returned to the MCT.

3.4.2.3. SPERRY UNIVAC 9200/9300 Subsystem

The 9200/9300 subsystem is handled by ICAM via the REM1 program, which performs the function of making the 9200/9300 subsystem an emulator of the 1004 card processor. The 9200/9300 subsystem as a remote terminal is handled by the 1004 RDH. For further details, see the discussion on the 1004 card processor in 3.4.2.2.

3.4.2.3.1. Reference Documents

Refer to the current versions of the following manuals for a more detailed description of the 9200/9300 subsystems:

- 9200/9300 remote communications manual, UP-7607
- 9200/9300 systems halt display manual, UP-7719

3.4.2.3.2. Operational Considerations

When using the 9200/9300 subsystem, note the following items.

- Restrictions
  - When emulating the 1004 card processor, the 9200/9300 subsystem cannot be multiplexed or multidropped.
  - The only legal auxiliary device function is:

    Output to the card punch.

- Establishing 9200/9300 Subsystem Connection

To establish connection from a 9200/9300 subsystem, use the following operating procedure:

1. Load the REM1 program from the card reader.

   If emulating a 1004 card processor with a DLT-3, the REM-1 deck is immediately followed by a site identification card.

   A 9200/9300 input deck may follow immediately behind the site-id card. If there is no 9200/9300 input deck, five or six blank cards should follow the site-id card as padding for the card reader.

   Establish the physical connection by dialing either from the 90/30 system to the remote 9200/9300 or from the remote 9200/9300 subsystem to the 90/30 system.

2. The 9200/9300 subsystem operator sets the MEMORY ADDRESS keys to 05_{16} and presses the OP REQUEST key to send the READY message to the host processor. This message is sent at 5-second intervals until acknowledged. See the discussion of the 1004 terminal for the method of handling the ready and site-id (3.4.2.2.2).

- Reading Cards

To read cards on the 9200/9300 subsystems, use the following procedure:

1. Ready the cards in the card reader. The card deck must terminate with an end-of-deck card (7 and 8 multiple punch in columns 1 and 2).

2. Set the MEMORY ADDRESS keys to 06_{16}.

3. Press the OP REQUEST key to start reading cards. The cards are read until the end-of-data card is encountered.

If the input deck was initially loaded immediately following the site-id card, only steps 2 and 3 need be performed to read the cards.

- 9200/9300 Subsystem Special Input

The operator can send special input messages to the host processor. They are identical to those that can be sent by the 1004 card processor. Refer to 3.4.2.2.2 for a definition of these inputs and how they are handled.

The only difference between the 1004 card processor and the 9200/9300 subsystem is the method in which the operator generates the special input.

To send the special input, use the following message procedures:

1. Set the MEMORY ADDRESS keys to the proper hex value.
2. Press the OP REQUEST key to send the message.

The MEMORY ADDRESS settings for each message are as follows:

Message	Hex Value Memory Address
READY	05
READ	06
HALT	07
HALT, GO VOICE	08
ABORT PRINT	09
ABORT PUNCH	0A
OFFLINE	0C
END READ	(Sent by remote 9200/9300 hardware)

- 9200/9300 Subsystem Special Output

  The one special output that the RDH can send to the 9200/9300 subsystem is the HALT message. It is handled as described in 3.4.2.2.2.

- 9200/9300 Subsystem Console Display

  Table 3-31 provides a list of the codes used to inform the 9200/9300 operator of the status of the remote 9200/9300 subsystem.

Table 3-31. 9200/9300 Subsystem Status Codes

Display	Reason	9200/9300 Operator Action
6601	HALT from host	Press START switch to continue. Send READY to resume communications.
6602	HALT from 9200/9300	
6603	HALT,GO VOICE from 9200/9300	
6604	Carrier lost on input line terminal	Press START switch to request retransmission.
6605	Nonoperational control unit or channel, or offline	Correct problem. Press START switch to continue.
6606	OFFLINE from 9200/9300	Start program (REM1) and send READY message to resume communication.
6607	Nonoperational input line terminal	Check setting of input line terminal switches. Press START switch to continue.
6610	SEND command to line terminal of DCS rejected	Command has been issued five times. Press START switch to continue. To cancel, key in a nonzero in location 4.
6611	Look for SYNC command to line terminal rejected	Perform action described for 6610.
6612	SENSE command rejected	
6613	TURN-OFF rejected	
6615	SENSE command for input rejected	
6618	SENSE information not expected for output	
6619	SENSE information not expected for output	
6620	Initial TURN-ON to line terminal of DCS not accepted	If DCS is offline, place DCS online. Press START switch to continue.

3.4.2.4. IBM 2780, IBM 3741, SPERRY UNIVAC UDS 2000

These terminals are designed to transfer large quantities of batch-type data over voice-grade facilities. These terminals are available in the following models.

IBM 2780	IBM 3741*	UDS 2000*
Card reader and printer	Diskette	Diskette
Card reader, punch, and printer	Diskette with printer	Diskette with printer
Printer only		
Card reader and punch		

The IBM 2780 and IBM 3741 can operate with any of three code structures – EBCDIC, ASCII, and TRANSCODE, as well as EBCDIC transparency.

UDS 2000 does not operate in ASCII and TRANSCODE codes.

3.4.2.4.1. Reference Documents

Refer to the appropriate IBM publication for a more detailed description of the IBM 2780 and IBM 3741. For UDS 2000 description, refer to UDS 2000 system reference, UP-8557 (current version).

3.4.2.4.2. Operational Considerations

These terminals cannot be multidropped or multiplexed on the same line as a SPERRY UNIVAC Series 90 System. When the terminal is sending, no output can be received until input is complete, unless an RVI sequence is transmitted to the IBM terminal.

Refer to the appropriate IBM publication for operational considerations. For the UDS 2000, refer to UDS 2000 operator reference, UP-8555 (current version).

**Both devices contain a keyboard display for preparing diskette input.*

3.4.2.4.3. BSC Software Support

The BSC RDH provides special software support for the remote terminal as discussed in the following paragraphs. It is a batch-oriented handler capable of operating in three modes:

1. Terminal Mode

   Allows communications between the SPERRY UNIVAC 90/30 System and an IBM 2780, IBM 3741, or UDS 2000 terminal, handling all hardware requirements of the remote terminal.

2. IBM 2780 Emulation (slave mode) RDH

   Allows the 90/30 system to emulate an IBM 2780 terminal, permitting communications between the 90/30 system and a host computer.

3. Generalized BSC mode RDH

   Allows communications between the 90/30 system and a remote CPU or between the 90/30 system and any terminal which adheres to the standard BSC line protocol. In this mode, no unique hardware functions for any specific terminal are performed.

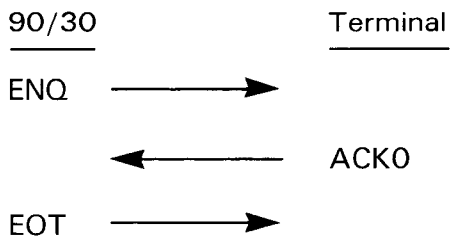
In addition, if the generalized BSC mode is chosen, the user may specify the use of a single large line buffer or the use of two smaller line buffers which are toggled.

When output is active, the RDH expects to receive raw text data from the user. When input is active, the user receives raw text data from the RDH. The RDH performs the following functions for the user:

- Line Contention

  If output has been posted by the user, the RDH attempts to gain control of the line for transmission of the text. If an input MCT has been posted, the RDH will accept line bid from the remote site.

  If an input MCT has been posted and the TN#MCBCN flag has been set in the TN#MCAD field, the RDH will transmit a special connection sequence to the remote terminal. This sequence is:



  It is required by certain terminals before they are able to bid for the line.

- Translation

  All output data is translated from EBCDIC to the specified line code (ASCII or TRANSCODE); input text is translated from line code to EBCDIC before it is passed to the user. No translation occurs when an EBCDIC line is specified.

- Build Envelope

  Upon receipt of input text from the user, the proper control characters are inserted before and after each record before transmission; these envelope characters are stripped from input text before it is passed to the user.

- Blocking of Data

  Where applicable, the RDH takes user-supplied output records and builds a data block, according to user specifications (record length, block length, number of records per block), for transmission to the remote site; when an input block is received, it is unblocked into individual records for the user.

- Device Selection

  During terminal mode operation, the RDH builds the proper escape sequence to select the device you specified in the auxiliary device field. In a slave mode, a selection escape sequence received from the host is detected and the user is notified whether the input is to be printed or punched.

- DICE Processing

  During terminal mode operation, the RDH converts user-supplied DICE sequences in the output text to hardware escape sequences to control vertical form spacing at the terminal. The RDH always expects the DICE sequence to be the last four characters in a network buffer. When passing input data to the user, the RDH inserts the standard batch DICE end-of-line sequence, unless the user has selected the no-DICE option. When operating in a slave mode, the RDH converts any escape sequences within the input text to DICE sequences to central printing at the 90/30 system. During generalized BSC mode operation, DICE sequences function as end-of-line sentinels only.

- Handshaking

  The RDH controls all handshaking with the terminal or host computer necessary to control input and output. No user intervention is necessary.

- Error Recovery

  The RDH controls all error recovery during input and output. The user is not notified of an error until the RDH has exceeded the specified retry count.

- LINE Disconnect

  If the user desires a disconnect sequence (DLE EOT) to be transmitted to the terminal when input or output is completed, set the TN#MCBDC flag in the TN#MCAD field of the related MCT. The user must issue a line release upon notification that input or output has terminated (not available on UDS 2000).

- Line buffers

  The user may choose to use either a single large line buffer in which an entire block of data is sent or received, or two smaller line buffers that are toggled during transmission and receipt of text data. Buffer toggling is available when operating in generalized BSC mode only.

  - Single line buffer

    The line buffer defined by the user when the network is generated must be large enough to hold the longest block of data that can be sent or received. When a single line buffer is used, it is possible to transmit more than one network buffer in a single block.

  - Toggled line buffers

    When using toggled line buffers, the user is restricted in that the contents of only a single network buffer can be sent or received in one transmission. Note that the blocked network buffer feature may be utilized.

The user is responsible for defining to the RDH the format of the data he expects to send and receive. The following specifications are fixed for a given line and terminal.

- Record Length

  The maximum number of characters that can be transmitted in a single record. Set in TERM proc for all modes.

- Block Length

  The maximum number of characters, including all data, device selection, vertical forms control, and end-of-record characters, that can be transmitted in a single block. Set in TERM proc. Not referenced when using blocked network buffers.

- Number of Records per Block

  The maximum number of records that can be transmitted in a single block. Set in TERM proc. Not referenced when using blocked network buffers.

- Transparent Input Record Length

  The fixed length of all transparent records received in multiple record blocks. Set in TERM proc.

- No DICE Option

  No DICE sequences are supplied in input network buffers. Set in TERM proc.

- Single Line Buffer/Toggled Line Buffer

  Type of line buffering to be employed. Set in LINE proc.

- Line Buffer Length

  Length of line buffer to be generated if single line buffering technique is used. Set in LINE proc.

The following specifications are active for a given input or output sequence only.

- Normal/Transparent Output

  If the special function field in the MCT has been set to indicate transparent data, the RDH performs all functions necessary to transmit the data in the transparent mode. If not set, output is sent as normal data.

The user should note that the BSC RDH supports all functions as described in 3.4.

The user should be aware of the following restrictions when operating with the BSC RDH:

- Blocked network buffers and toggled line buffers are available when operating in generalized BSC mode only.

- When receiving multiple record transparent input in any mode, each record must be of fixed length, as specified in the TERM proc.

- The line buffer supplied by the user when using the single line buffer technique must be large enough to contain the largest data block, including all control characters. If the end of the line buffer is reached before the end of the network buffer, when the user is sending blocked output network buffers, the data will be truncated. When receiving input data, if the end of the line buffer is reached before the end of the data, part of the message is lost.

- Network buffers supplied by the user on input must be large enough to hold the largest input data record (if single record network buffers are used) or input data block (if blocked network buffers are used). If the network buffer is not large enough, a format error status is returned to the user and input is terminated.

- No DICE sequences can be used in transparent output data. None are supplied in transparent input except when operating in IBM 2780 emulation (slave) mode.

- No DICE sequences can be used in blocked network buffer data and none are supplied in input. Text received from the user when using blocked network buffers is assumed to be raw text data, with no interrecord control characters. Text sent to the user has the same format.

- When toggled line buffers are used, the data for one transmission (a block) must be contained in a single network buffer (blocked or single record).

3.5. WORKSTATION SUPPORT

This subsection discusses workstation characteristics relating to ICAM.

3.5.1. Reference Documents

For a more detailed description of workstations, refer to the current version of interactive services commands and facilities, user guide/programmer reference, UP-8845.

3.5.2. Operational Considerations

Workstation start-up procedures are described in the current version of the ICAM network definition and operations user guide, UP-8947.

Index

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD