UNISYS

BTOS

Burroughs Multipoint Communications Service (BMULTI)

Operations and Programming Guide

Relative to Release Level 7.0

July 1[illegible]

Priced Item 1212727

UNISYS

BTOS

Burroughs Multipoint Communications Service (BMULTI)

Operations and Programming Guide

Relative to Release Level 7.0 — July 1987

Priced Item — 1212727

Unisys believes that the software described in this manual is accurate, and much care has been taken in its preparation.

The customer's attention is drawn to the provisions of the Trade Practices Act 1974 (as amended) ('the Act') which imply conditions and warranties into certain contracts for the supply of goods and services. Where such conditions and warranties are implied Unisys liability shall be limited (subject to the provision of Section 68A of the Act) to the replacement or repair of the goods or the supply of equivalent goods.

The customer should exercise care to assure that use of this manual and the software will be in full compliance with the laws, rules and regulations of the jurisdiction in which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

Correspondence regarding this publication should be forwarded, using the Product Improvement Card at the back of this manual, or remarks may be addressed directly to Unisys Corporation, Corporate Product Information East, Building C, Township Line and Union Meeting Road, Blue Bell, PA 19424 U S America.

About This Manual

The Burroughs Multipoint Communications Service (BMULTI) allows a BTOS system (B 26, B 27, B 28, B 38 or XE 520) to communicate with larger Burroughs systems using the Burroughs multipoint protocol.

Purpose

This manual enables system administrators and programmers to install BMULTI, operate its line monitor, and write programs that use its data communication services.

Scope

This manual describes installing, configuring, operating, and troubleshooting BMULTI. It also supports the writing of programs that use BMULTI's programmatic interfaces.

Audience

The audience of this manual is system administrators who wish to install and operate BMULTI and programmers who want to write applications that use BMULTI's services. System administrators can use sections of this manual to guide operators (non-programmers) in configuring and installing BMULTI.

Prerequisites

Programmers should be familar with BTOS and the programming resources available from Unisys including the BTOS editor, debugger, and linker, as well as the languages and compilers supported by BTOS.

How To Use This Manual

Where to Find the Information You Need

To:	You Must:	Read Sections:
Run an application such as DTS, MT983)	Install, configure, and load BMULTI	2 3 C H I
Install BMULTI	Install, configure, and activate BMULTI	2 3 C H I
Configure BMULTI	Obtain configuration values and use the Configurator to enter	2 3 C H I
Configure BMULTI dynamically	Use the dynamic configuration feature of the Configurator	3
Deinstall BMULTI		2
Program BMULTI	Use programming interfaces	6 7 8 9 10 A C E
Create Applications	Use programming interfaces	6 7 8 9 10 A D E I
Run earlier BMULTI applications	Reconfigure BMULTI Replace BMOpen call	3 10 A H
Troubleshoot BMULTI	Refer to text and error messages	3 4 5 F G H
Monitor BMULTI	Use status and line monitors	4 5
Monitor BMULTI line	Use line monitor	2 5
Monitor BMULTI	Use status monitor status	4
Understand BMULTI Concepts		5 6 7 8 Glossary
Debug BMULTI programs		7 8 F G H

References

BTOS Systems B-NET Administrator's Guide

BTOS Systems Context Manager Reference Manual

BTOS Systems Operating System Reference Manual

BTOS Systems Programmer's Guide

BTOS Systems Standard Software Operating Guide

XE 520 System Programmer's Guide

Contents

About This Manual v
Purpose v
Scope v
Audience v
Prerequisites v
How To Use This Manual vi
References vii

Section 1: Overview 1-1
BMULTI Features 1-1
Memory, Hardware, and Operating System Requirements 1-2
Using Previous Versions of BMULTI 1-3

Section 2: Software Installation 2-1
Installation Procedure 2-1
Invokin BMULTI 2-2
Invoking BMULTI Automatically 2-2
Deinstalling BMULTI 2-2
Files on the Installation Disk 2-3

Section 3: Configuring BMULTI 3-1
Configuration Worksheet 3-1
Configuration Parameters 3-2
Using Configuration Files from BMULTI 5.0 and Earlier 3-4
BMULTI Configurator 3-5
Entering the Configurator 3-5
Entering Values in the Configurator 3-5
Leaving the Configurator and Activating BMULTI 3-7
Dynamically Configuring BMULTI 3-7
Opening Disk Files from the Configurator 3-8

Section 4: The Status Monitor 4-1
Status Monitor Labels 4-3
Displaying Additional Addresses 4-4
How to Purge an Address 4-4

Section 5: The Line Monitor 5-1
Entering the Line Monitor 5-1
Function Key Menu and Line Monitor States 5-2
File Name 5-3
Ring Buffer 5-4
Screen Messages 5-4
Changing the Buffer Size 5-4
Viewing Data from BMULTI 5-4

Overview

BTOS workstations and XE 520 systems use Burroughs Multipoint Communications Service (BMULTI) to communicate with larger Burroughs systems. BMULTI provides the support that applications need to make BTOS workstations function like standard Burroughs terminals.

Some BMULTI applications are:

- MT 983 Emulator
- Data Transfer Service (DTS)
- Burroughs Terminal Emulator (BTE)

BMULTI is transparent to users of these terminal emulators.

BMULTI Features

Bmulti supports the following:

- Up to 64 addresses, each handling messages of up to 4096 bytes (including protocol control characters). The number of active addresses you should use depends on the applications and on the response time you need.
- Synchronous transmission at 110 to 9600 bits per second
- Asynchronous transmission to 38400 bits per second (transmission rates above 19200 bits per second must be run on the Intelligent Data Communications Slice.)
- Normal poll, normal select, group poll, fast select, group select, broadcast select, and multipoint contention
- Several transmission numbering options
- 50 ms host timeout when all applications in the network are linked with BMULTI 6.0 and higher.
- Deinstallation
- Context Manager
- Access from remote node via B-NET
- User configuration of delays for Clear-to-Send, Transmit-to-Receive, and Request-to-Send-Hold

- Configuration to answer polls of up to 64 inactive addresses
- Protected Mode
- BTOS II
- Dynamic configuration of BMULTI
- A status monitor that allows you to examine and store data traffic
- A line monitor

Memory, Hardware, and Operating System Requirements

- BTOS workstations must have at least 512 K memory.
- BMULTI requires at least 36 K of user memory. In addition, the object modules that must be linked with an application require additional memory, which varies depending on which programmatic interface is used. XE 520 cluster or terminal processors cannot use BMULTI without a memory expansion board.
- The line monitor requires a minimum of 108 K memory.
- BMULTI can run on any B 25 family workstation and on either the cluster processor or the terminal processor of an XE 520. Hardware requirements are further explained in Appendix C.
- BTOS 8.0 operating system is required for BTOS workstations and BTOS masters.
- MS 8 or higher is the recommended operating system for the XE 520.

The following table describes BMULTI Memory Requirements, depending on which interface is used. The interfaces are described in Sections 9 and 10.

Table 1-1 **BMULTI Memory Requirements**

Interface	Memory Required
Multitasking Low-Level	5400 bytes
Multitasking High-Level	9900 bytes

Using Previous Versions of BMULTI

Applications linked with this release of BMULTI.lib do not run with earlier versions of the BMULTI system service. The system service supports only version 6.0 of the older libraries but does not support the BmQuery and BmIdentify procedural interfaces in this library. Any application using these interfaces must be rewritten using the BmGetStatus procedural interface, then recompiled and relinked using the BMULTI 7.0 library. See Section 3 for information about updating configuration files created with previous versions of BMULTI. In addition, if you run applications that are not linked with this release of BMULTI, you cannot use some of its options.

BMULTI supports applications that were written using the Multiple-Task Interface (MTI) and Single-Task Interface (STI) from previous versions of BMULTI. When you create new applications or modify existing ones, use the High-Level Interface (HLI) and Low-Level Interface (LLI) instead of the older interfaces. For more information, see Appendix F.

"Freezing" when Receiving Data 5-5
Exiting Receive 5-5
Loading a File to the Ring Buffer 5-6
Save 5-7
Clearing the Buffer 5-7
"Jumping" and Scrolling Through Data 5-7
Troubleshooting with the Line Monitor: A Sample Walkthrough ... 5-10

Section 6: Basic BMULTI Concepts 6-1
Recommended Interfaces 6-1
Linking Applications with BMULTI.lib 6-2
Commands, Reports, and Requests 6-2
Virtual Addresses (VADs) 6-3

Section 7: BMULTI State Machine 7-1
Reserving a Station Address 7-3
Online, Offline, and Idle 7-3
Transmitting 7-4
Receiving 7-4
Idle and Abort 7-4
Receiving Fast, Group, and Broadcast Selects 7-5
Terminating a Communication Session 7-5
Report Queue 7-5
BMULTI States 7-5
Offline 7-6
Idle 7-6
Transmit Ready 7-8
Transmitting 7-10
Transmitting and Receive Ready State 7-12
Transmit and Receive Ready State 7-13
Receiving State 7-14
Receive Ready State 7-15
Receiving and Transmit Ready State 7-17

Section 8: Protocol Description 8-1
Asynchronous Data Communication 8-1
Synchronous Data Communication 8-1
Data Accountability 8-2
Alternating Transmission Numbering 8-2
Sequential Transmission Numbering 8-2
No Response Timeout 8-3
Idle Line 8-3

Section 9: High-Level Procedural Interface 9-1
CloseBMULTI 9-3
Description 9-3
Procedural Interface 9-3

OpenBMULTI 9-3
Description 9-3
Procedural Interface 9-3
ReadBMULTI 9-4
Description 9-4
Procedural Interface 9-4
ResetBMULTI 9-5
Description 9-5
Procedural Interface 9-5
SetOptionBMULTI 9-5
Description 9-5
Procedural Interface 9-5
WriteBMULTI 9-6
Description 9-6
Procedural Interface 9-6
SelectBMULTI 9-7
Description 9-7
Procedural Interface 9-7
SetXlatModeBMULTI 9-8
Description 9-8
Procedural Interface 9-8

Section 10: Low-Level Interface 10-1
B-NET Support 10-1
BmCommand 10-1
Description 10-1
Procedural Interface 10-2
BmOpenII 10-3
Description 10-3
Procedural Interface 10-3
BmReport 10-4
Description 10-4
Procedural Interface 10-4
BmReportTimeout 10-5
Description 10-5
Procedural Interface 10-5
BmReportWait 10-5
Description 10-5
Procedural Interface 10-5
BmGetStatus 10-6
Description 10-6
Procedural Interface 10-6
Common Information Buffer Format 10-7
Station Record Format 10-8
Description 10-9
Procedural Interface 10-10

Appendix A: Sample Programs A-1
COBOL Echo Program (Using High-Level Interface) A-1
BASIC Echo Program (Using High-Level Interface) A-4
FORTRAN Echo Program (Using High-Level Interface) A-6
Pascal Terminal (Using High-Level Interface) A-8
Pascal Terminal (Using Low-Level Interface) A-15
USASCII Code Charts A-1

Appendix C: Hardware Requirements C-1

Appendix D: Language Configuration D-1
BASIC (BasGen.asm) D-1
COBOL (COBOLGen.asm D-2
FORTRAN (ForGen.asm D-3

Appendix E: BTOS Request Codes for BMULTI E-1

Appendix F: Troubleshooting Notes F-1
Rebooting the Master F-1
Receive Buffer Overflow F-1
B-NET Support F-1
Earlier BMULTI Versions F-2
BmOpenII, VADs, and Existing Applications F-2
VADs and Sophisticated NDLs F-2
Choosing the Channel to Use with the Data Comm Expander Module F-2
Response Time F-3
Configurable Delays F-3
Transmission Blocks F-3
Configuration Parameters F-4
Status Code 32786, "Transfer Xmt Buffer Command Denied" F-4

Appendix G: Status Codes Generated by Low-Level Interface . G-1

Appendix H: Host/NDL Requirements H-1
Mainframe/NDL Timeout Values H-1
Virtual Addresses and Network Definition Languages H-1

Appendix I: BMULTI Configuration Worksheet I-1

Appendix J: Examples of Shift-In and Shift-Out Action J-1
BTOS Receiving Format J-1
BTOS Transmitting Format J-2

Appendix K: Translation Tables K-1
Translation Template BTOS USA Standard Character Set to T 27 Version 1 (USA) Character Set K-3

BTOS Receive Translation Table K-3
BTOS Transmit Translation Table K-4
Translation Template BTOS Canadian Standard Character Set to T 27 Version 1/A (Canada) Character Set K-5
BTOS Receive Translation Table K-5
BTOS Transmit Translation Table K-6
Translation Template BTOS United Kingdom Standard Character Set to T 27 Version 2 (UK) Character Set K-7
BTOS Receive Translation Table K-7
BTOS Transmit Translation Table K-8
Translation Template BTOS Netherlands Standard Character Set to T 27 Version 2 (UK/Belgium/Italy) Character Set K-9
BTOS Receive Translation Table K-9
BTOS Transmit Translation Table K-10
Translation Template BTOS South African Standard Character Set to T 27 Version 2 (UK/Belgium/Italy) Character Set K-11
BTOS Receive Translation Table K-11
BTOS Transmit Translation Table K-12
Translation Template BTOS German Standard Character Set to T 27 Version 5/A (Germany) Character Set K-13
BTOS Receive Translation Table K-13
BTOS Transmit Translation Table K-14
Translation Template BTOS French Standard Character Set to T 27 Version 43 (French Word Processor) Character Set K-15
BTOS Receive Translation Table K-15
BTOS Transmit Translation Table K-16

Glossary 1

Index 1

Illustrations

3-1	**BMULTI Configuration Parameters**	3-1
3-2	**Configurator Function Key Menu**	3-6
4-1	**A Sample BMULTI Status Monitor Screen**	4-2
5-1	**Initial Line Monitor Screen**	5-2
5-2	**Initial Function Key Menu**	5-3
5-3	**Local Function Key Menu**	5-3
5-4	**Receive Function Key Menu**	5-3
5-5	**Sample Receive Session**	5-5
5-6	**Screen after Exit from Receive**	5-6
5-7		5-8
5-8		5-9
7-1	**BMULTI States**	7-2
8-1	**Specific Polling**	8-4
8-2	**Group Polling**	8-6
8-3	**Selection**	8-9
8-4	**Fast Select**	8-11
8-5	**Broadcast Select**	8-13
8-6	**Group Select**	8-15
8-7	**Multipoint Contention Mode**	8-17
B-1		A-1
B-2	**USA Standard Code for Information Interchange (USASCII)**	A-2
C-1	**RS-232C Signals in Operation**	C-2

Tables

1-1	**BMULTI Memory Requirements**	1-2
6-1	**BMULTI.lib Modules and Procedure Calls**	6-2
9-1	**Format of the Application Status Block**	9-2

Software Installation

This section explains how to install BMULTI so that you can run BMULTI applications on your system. Here is the general procedure for installing BMULTI:

1 Create a configuration file, as described in Section 3.

2 Perform the steps under the appropriate heading for the way you want to activate BMULTI: "Installing the BMULTI System Service" or "Automatically Installing the BMULTI System Service". XE 520 systems can use the automatic installation procedure (through a JCL file).

Installation Procedure

1 Insert the product distribution disk, labeled *BTOS Poll Select, Disk 1 of 2*, into floppy disk drive [f0].

2 If you are installing on an XE 520, type **XE Software Installation** at the command line. Otherwise, type **Software Installation**.

3 Press RETURN. The following form appears:

```
Software Installation
   [Command file]
   [Files to]
   [Confirm?]
   [Install file]
```

If you are installing on an XE 520, make sure your entry on the [Command File] line is correct. Leave the [Files to] parameter blank, because certain BMULTI files must be in the default <Sys> directory.

4 Press GO and follow the directions on the screen.

5 Remove the disk from the floppy disk drive.

6 Reboot the system when installation is completed.

Before operating BMULTI, you must edit the default configuration file as described in Section 3.

Invoking BMULTI

Type **Install BMULTI** at the Command line and press GO.

If your system encounters a fatal error and the signon form appears, it may indicate that your master is not running.

BMULTI will not install on systems with a four-port expansion module if the data communications server was not installed and BMULTI was configured with channel 1A, 1B, 1C, 1D, 2A, 2B, 2C, or 2D.

Invoking BMULTI Automatically

To automatically install BMULTI whenever you turn on or reboot your BTOS system, add the following line to the appropriate JCL file. In this line, optional text is italicized:

```
$RUN [Sys]<Sys>BmZip.run, <configuration file>
```

The appropriate JCL files in which to include this line are as follows:

SysInit.JCL for workstations

InitCpnn.JCL for cluster processors (nn is the Cp number)

InitTpnn.JCL for terminal processors (nn is the Tp number)

Deinstalling BMULTI

The command Deinstall BMULTI frees the memory and physical channel that the BMULTI service is using, reversing the effect of the command Install BMULTI. The Deinstall BMULTI command operates under these conditions only:

- The operating system is multipartition.
- The command is executed from the workstation on which BMULTI was installed.
- The system is not an XE 520.
- The command is executed from the primary application partition.

- No applications (station addresses) are active. For example, BTE cannot be running when you deinstall BMULTI.
- All systems services that are clients of BMULTI are deinstalled.
- The Context Manager is deinstalled.

To deinstall BMULTI:

1 Make sure your workstation is at the Executive level with your screen showing the prompt "Command."

2 Type **DeInstall BMULTI** and then press GO.

If no error message is displayed, BMULTI was successfully deinstalled. These are the error messages that you will encounter if BMULTI was not deinstalled:

Status Code	Hex	Meaning
32798	801E	BMULTI was not deinstalled because operating system is a single-partition version
32799	801F	BMULTI was not deinstalled, either because the DeInstall utility was not executed in the primary application partition (Context Manager installed) or because the DeInstall utility was not executed on the machine on which BMULTI is installed.
32802	8022	BMULTI was not deinstalled because stations are active.

Files on the Installation Disk

The software installation procedure should copy the following files from the release disk into the <Sys> directory of your hard disk.

Volume POS6.0-1

Directory name	Sectors	Default protection
SYS	3	15
B25PS7	3	15

Files on the disk:

<Sys>BmLineMonitor.run
<Sys>Request.F.sys
<Sys>BmMonitorFont.B27
<Sys>BmMonitorFont.B25
<B25PS7>BMULTI.run
<B25PS7>BmZip.run
<B25PS7>BmUnZip.run
<B25PS7>BmStatus.run
<B25PS7>BmFileEdit.run
<B25PS7>BMULTI.lib

Configuring BMULTI

Before you can use BMULTI, it must be configured for your system. This section:

- defines the parameters used in configuration.
- explains how to determine the correct values for these parameters.
- explains how to use the Configurator to enter those values.
- explains how to upgrade a configuration file created with release levels 4.0 and higher of BMULTI.

Configuration Worksheet

The worksheet below is for your records. If, as a system administrator, you want to give an operator (non-programmer) a list of installation parameters to use, fill out the worksheet supplied in Appendix I. To accomplish installation, operators should read "BMULTI Configurator" in this section.

Figure 3-1 **BMULTI Configuration Parameters**

Screen Message	Preset Value	Change to
Group Poll Address	dv	_____
Group Select Character	r	_____
Sync or Async	A	_____
Channel	0B	_____
Baud Rate	09600	_____
Transmission Numbering	0	_____
RTS-CTS delay	0	_____
XMT-RCV delay	0	_____
RTS HOLD delay	0	_____
Anything Downstream?	N	_____
TDI on IDS?	N	_____
Number of Stations	16	_____
Buffer size	2048	_____

Configuration Parameters

Group Poll Address The group poll address to be used by all BMULTI stations. A cluster system or any system accessing BMULTI over B-NET can have only one group poll address. This must be any two ASCII characters between 020h and 07Fh, as listed in Appendix B. If group poll is not used, any address not polled may be used.

Group Select Character The character recognized as the group select character to be used by BMULTI for all its stations. A cluster system or any system accessing BMULTI over B-NET can have only one group select character. This must be any ASCII character between 020h and 07Fh, as listed in Appendix C. If you do not use group select, you may use any character not already in use as a poll or select character.

Sync or Async Enter **S** (Synchronous) and the modem supplies clocking. Enter **A** (Asynchronous) and the workstation supplies clocking. Always enter **A** for a TDI line.

Channel On a BTOS workstation and an XE 520 cluster processor, this is either A or B. On an XE 520 terminal processor, this is A, B, C, or D. On workstations with a Data Comm Expander (DCX) module, the channel can be A, B, 0A, 0B, 1A, 1B, 1C, 1D, 2A, 2B, 2C, or 2D.

For BMULTI to take control of a specified channel, that channel must not be already under the control of another program. On an XE 520, no "ASYNC <channel number>" statement may refer to that channel in the appropriate configuration file (Cpnn.cnf for the cluster processor and Tpnn.cnf for the terminal processor, where nn is the cluster or terminal processor number). Make certain that the channel you assign is not assigned to your system's spooler.

Baud Rate Transmission speed in bits per second. Synchronous transmission must be 110, 150, 300, 600, 1000, 1200, 1800, 2000, 2400, 4800, or 9600. The XE 520 cannot use 110 and 150 baud. Asynchronous speeds are the same except that the maximums are:

Workstation Type	Maximum bps
B 26	19200
B 27	19200
B 28	19200
4-port	9600
IDS	38400
XE 520	19200

Note: "4-port" refers to workstations with four-port expansion modules.

Xmno option [0..5] 0, 1, 2, 3, 4, or 5 Each of these numbers represents a transmission numbering scheme:

Number	Transmission Numbering Scheme
0	no transmission numbers
1	alternating zero and one scheme
2	alternating @ and A (TD830 compatibility)
3	n modulus 10 (0 through 9, wrapping around)
4	n modulus 100
5	n modulus 1000

See Section 8 for more information.

RTS-CTS delay [0..255] The clear-to-send delay in milliseconds. After you have turned on request-to-send, BMULTI waits the amount of time set here before looking for clear-to-send from the modem. If clear-to-send is not on when the timer expires, BMULTI waits until it goes on before it transmits.

XMT-RCV delay [0..255] The receive delay in milliseconds. When BMULTI turns off request-to-send after a transmission, it waits this amount of time before examining incoming data. This delay is normally not zero when using Burroughs Two-wire Direct Interface (TDI).

RTS Hold Delay [0..255] The number of milliseconds that BMULTI keeps request-to-send on after the end of a transmission. This delay is used with some older modems or to have the host system modem keep Data Carrier Detect on long enough to ensure that the host receives the transmission.

Downstream Station If you enter **Y** (yes), BMULTI does not reply to a group poll when secondary receive data is on. If you enter **N** (no), BMULTI ignores secondary receive data when determining its response to a group poll.

Enter **Y** if the workstation running BMULTI is in the midst of a concatenated string of terminals communicating through a single modem.

Enter **N** if the workstation is the only "terminal" connected to its modem, if it is the last terminal on a concatenation string, or if you are using TDI. This parameter does not apply to XE 520 systems, which always act as if the value were sent to **N**.

TDI on IDS? (Two-wire Direct Interface) Enter **Y** (yes) if you are directly connecting your workstation to a data communication network through a TDI physical interface. Enter **Y** only if you are using a device that allows direct connection to a TDI network such as an Intelligent Data Comm Slice (IDS). Enter **N** (no) if you are connecting your workstation to a data communication network through an RS-232 physical interface. Enter **N** if you are using a Data Communication Adapter (DCA). See "Using Configuration Files from earlier BMULTI versions."

Number of Stations [1..64] Enter the maximum number of addresses on this copy of BMULTI.

Buffer size [0..4096] Enter the size, in bytes, of the BMULTI Transmit/Receive buffer. The size specified will be the size of the receive buffer and the extended message transmit buffer.

Using Configuration Files from BMULTI 5.0 and Earlier

If you have configuration files created with release level 4.0, 5.0, or 6.0 of BMULTI, you must open them, make any new parameter selections, and close them using the BMULTI 7.0 configurator.

Note: The channel and Baud Rate parameters must be right-justified with no spaces in the field.

BMULTI Configurator

The Configurator is an editor that enables you to customize BMULTI for your system. You don't need to be a programmer to use the Configurator; however, you should be familar with the general operation of BTOS systems.

Before you begin, have Appendix I or Figure 3-1 handy, filled out by your system administrator.

Entering the Configurator

1 Type **Configure BMULTI** on the Command line and press RETURN. The following form appears:

```
Configure Bmulti
 [Configuration file] _____________________
```

2 The default configuration file is [Sys]<sys>BMULTIConfig.Sys. If you want to create a file with a different name, type the file name.

3 Press GO.

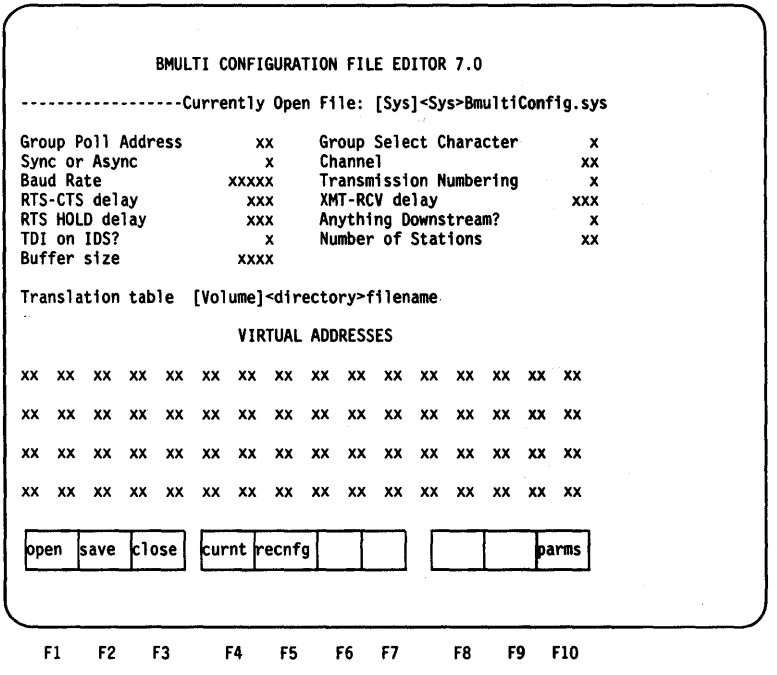
Entering Values in the Configurator

To enter the values given to you on the tear-out worksheet, use these guidelines:

- To move within a field, use the Right-Arrow and Left-Arrow keys.
- To move to the following field, press NEXT, RETURN, or the Down-Arrow key. To move to the previous field, use the Up-Arrow key.
- If you enter an unacceptable value, your workstation beeps and prevents you from leaving that field until you have corrected your error. If you cannot discover an acceptable value, enter the appropriate preset value from Figure 3-1.

When you have finished entering values, press GO. The bottom line of your screen shows ten boxes. Some of these boxes list commands that may be initiated by pressing one of the function keys on your keyboard.

Figure 3-2 **Configurator Function Key Menu**



To save the changes you have made on your disk, press SAVE, F2. To activate the changes you have entered, refer to the next heading in this section.

If you do not want to save the changes you have made, press CLOSE. To leave the Configurator, press FINISH and then press GO.

Leaving the Configurator and Activating BMULTI

To leave the Configurator and activate BMULTI, complete the following steps. To open disk files or dynamically change parameters, see the following headings in this section.

1 Press FINISH. You return to the Executive.

2 Type **Install BMULTI** and press GO.

If your screen displays status code 8401 or 32811, go on to steps 3 and 4. Otherwise, BMULTI is now installed on your system. You must reinstall BMULTI by repeating step 2 whenever you turn your system on.

3 Type **DeInstall BMULTI.** Press GO. If you receive an error message, see Section 2.

4 Type **Install BMULTI**. Press GO.

BMULTI is now activated on your workstation. Remember, you must type **Install BMULTI** every time you turn on your workstation, (unless you followed instructions in Section 2 to automatically complete this step).

Dynamically Configuring BMULTI

To reconfigure BMULTI after you have installed it, use the following procedure.

1 Enter the Configurator.

2 Press CLOSE. This closes the default disk configuration file.

3 Press CURRENT to display the parameters that are currently being used by BMULTI. (The parameters will be different if you previously dynamically configured BMULTI, representing the difference between the parameters of the BMULTIconfig.sys file that is opened automatically when you enter the Configurator and your previous dynamic changes.)

4 Press PARMS and make your changes.

5 Press the RECNFG function key to reconfigure BMULTI.

6 Press FINISH.

Be aware that the parameters that you entered are not saved to disk in a configuration file. They exist only in RAM and are current only as long as BMULTI remains installed.

Opening Disk Files from the Configurator

1 Press GO until the function key menu resembles Figure 3-2.

2 Press CLOSE.

3 Press OPEN.

OPEN is now lit. Your cursor is blinking in the upper right corner of your screen, beneath [Sys].

4 Overtype the name currently on your screen with the filename you want to create or alter. Use the space bar to erase extra characters.

5 Press OPEN.

6 To change the parameter values in your new file, follow the directions in "Entering Values in the Configurator."

The Status Monitor

Use the status monitor to determine which applications are using BMULTI, and how busy they are. You may also use the status monitor to purge an address, unless you are running the monitor from a remote node.

To activate this monitor, type **BMULTI Status.** Press RETURN. The screen displays:

```
BMULTI Status
  [Node name]
```

This feature enables you to monitor BMULTI running on a remote BNET node. Press GO; otherwise, enter the name of the node you want to monitor and press GO.

Figure 4-1 illustrates a sample screen. The screen is updated at one-second intervals. Short messages transmitted from a cluster station may be completed before the screen is updated, causing it to appear as if no transmittal has occurred.

The line displayed in dim reverse video is called the selected address.

Figure 4-1 **A Sample BMULTI Status Monitor Screen**

```
                    BMULTI STATUS MONITOR 0.0

VERSION:  R0.0.0000      INSTALLED:  At Master      ON CHANNEL:  B

# STATIONS IN USE:  16   NAK COUNT:  201           LINE:  Active

 #   ADDR  PROGRAM     USER      STATE     WS  RQs  TIME ON  SENT  RECD
---------------------------------------------------------------------
 01   1a   MT983 Em   Admin     Rcv Rdy   32   1      180   101   100
 02   1b   B 20 FT2   Desmond   Xmt Rdy   03   1     2880    45    45
 03   1c   DTS 2.0    Accntng   Idle      21   0      120    75    75
 04   1d              User      Rcv Rdy   41   1       60    20   120
 05   1e   InfoLink              Xmting    17   2     7200    95    95
 06   1f   PPT                   Rcv Rdy   31   1     1440     1    50
 07   1g   T 27 Em    Connie    Xmt/Rcv   51   2      240   201   200
 08   1h              Gene      Idle      62   1        5    10    10
 09   01   DOCxfer    WRITE2    Rcv Rdy   32   1       30    60    60
 10   02              Bernard   Xmt Rdy   03   1      480    55    45
 11   03                        Idle      21   0        1     0     0
 12   71   Infoview   Gerry     Rcv Rdy   41   1      180     5     1
 13   72   Infoview   Sftwre    Xmting    17   2      180     4     2
 14   73   Infoview   Pat       Rcv Rdy   31   1      180     3     3
 15   74   Infoview   Dvlmpt    Xmt/Rcv   51   2      180     2     4
 16   75   Infoview   Supprt    Idle      62   1      180     1     5
```

Status Monitor Labels

- **Version**: Lists the BMULTI release level.
- **Installed**: Shows type of workstation on which BMULTI is installed.
- **On Channel**: Indicates the channel on which BMULTI is running.
- **# Stations in Use**: Shows the number of workstations using BMULTI.
- **NAK Count**: The number of NAKs BMULTI sends in reply to messages. This helps you determine line quality. Not counted are: NAKs sent by BMULTI to selects or NAKs received by BMULTI.

- **Line**: Active or inactive. This indicates if the line was used in the previous second.
- **Addr**: Address.
- **Program**: Application, such as BTE or File Transfer.
- **User**: Logon ID.
- **State**: See Section 7 for more explanation.
- **Ws**: Workstation number, assigned by the operating system.
- **Rq**: Number of requests outstanding.
- **Time on**: Number of minutes since application was initiated.
- **Sent**: Number of messages transmitted since application was initiated.
- **Rcd**: Number of messages received since application was initiated.

Displaying Additional Addresses

If there are more addresses online than your screen can show, use the scrolling or paging keys to display the additional addresses.

How to Purge an Address

In the rare event that you need to purge (free) an address, you can easily do so while you are running the status monitor, unless you are accessing a remote B-NET node:

1. From the Executive, type **BMStatus**. Press GO.
2. The address line displayed in dim reverse video is called the selected address. Use the arrow keys to select the address you need to purge.
3. Press MARK. The selected address line changes to bright reverse video.
4. To purge the selected address, press DELETE. If you decide not to purge the marked address, press CANCEL.

The Line Monitor

The line monitor is a software facility that shows all data passing through BMULTI. Because some of this data may be proprietary, you may want to load the monitor from a floppy disk. If security is not an issue on your system, you may install the monitor during software installation.

When you use the line monitor, data from BMULTI is saved in a ring buffer (a ring buffer is one in which the newest information overwrites the oldest). The ring buffer size is configurable from 1K to 60K. The default size is 10K.

The line monitor looks for the file BMLineMonitorFont.BXX (where xx represents the type of workstation, e.g., 25 = B 25). If this file is not located, the appearance of line monitor data may be affected, however the data is not changed.

Using the monitor's menu, you can:

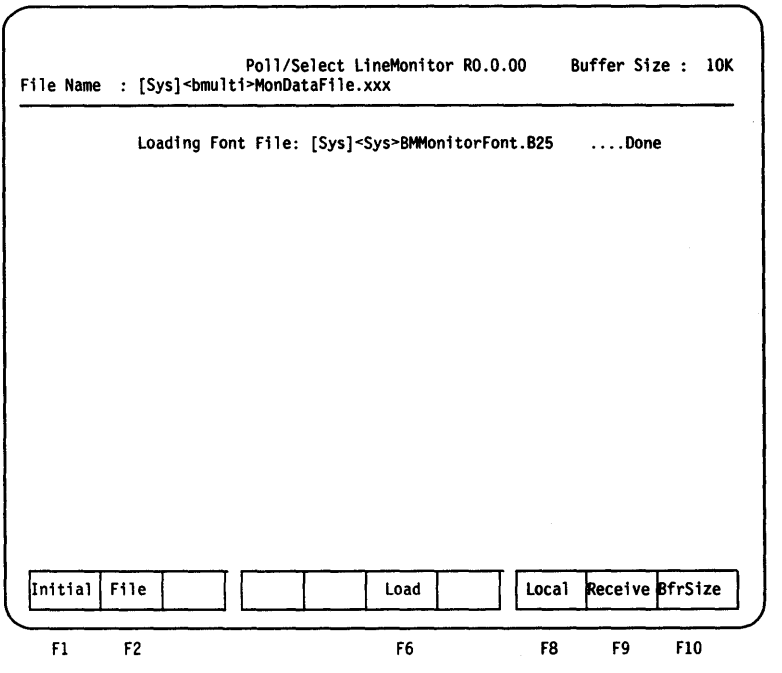
- "Freeze" the screen display while data is being received
- Save any portion of buffered data to a disk file
- Buffer up to 60K of data in memory
- Scroll through buffered data using the page and scroll keys, or the "jump" function
- Load and examine data that was previously saved by the line monitor

Entering the Line Monitor

To enter the monitor from the Executive, type **Monitor BMULTI** and press GO. When in the line monitor, you can access on-line help by pressing HELP. On-line help will give you a brief description of commands and types of error messages.

The initial line monitor screen resembles Figure 5-1.

Figure 5-1 **Initial Line Monitor Screen**



Function Key Menu and Line Monitor States

The line monitor has two states, local (offline) and receive (online). In the local state, the line monitor is logged off BMULTI and under operator control. In receive state, the line monitor is logged on to BMULTI. There are two function key menus in the local state: initial (Figure 5-2) and local (Figure 5-3). There is only one function key menu in receive state (Figure 5-4).

When you first enter the line monitor, the initial menu is displayed. As soon as you perform any line monitor function, either the local or the receive menu is displayed. A soft key is always highlighted; this indicates which menu is being displayed or which option has been chosen from the menu.

Figure 5-2 **Initial Function Key Menu**

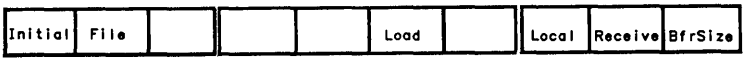
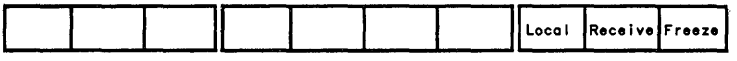


Figure 5-3 **Local Function Key Menu**

Figure 5-4 **Receive Function Key Menu**



To perform functions such as SAVE and JUMP, you must be logged off BMULTI. The softkey LOCAL is highlighted. If receive or freeze is highlighted, you are logged on to BMULTI. Press LOCAL to log off BMULTI and return to the Local state.

File Name

The default filename is [Sys]<sys>MonDataFile.XXX. This filename is automatically given to any data files you receive or save, unless you change the default filename or specify another filename when you load or save a file.

To change the default filename for a particular session:

1 Press FILE. The current default filename is displayed.
2 Type the name of the file that you want. To correct errors, press BACKSPACE or the left-arrow key.

Ring Buffer

Screen Messages

Information about the buffer appears on the top two lines of the screen. Here's an explanation of each caption:

Bytes in Buffer (Bytes in Bfr): Number of bytes currently in the buffer.

Buffer Size: The number of bytes allocated for the ring buffer.

The Buffer Position (Buffer Posn): The offset of the first byte of data visible in the display window. This number is zero relative; the first offset in the buffer is numbered 0, the second, 1, and so on.

Changing the Buffer Size

Note: Changing the buffer size clears the buffer.

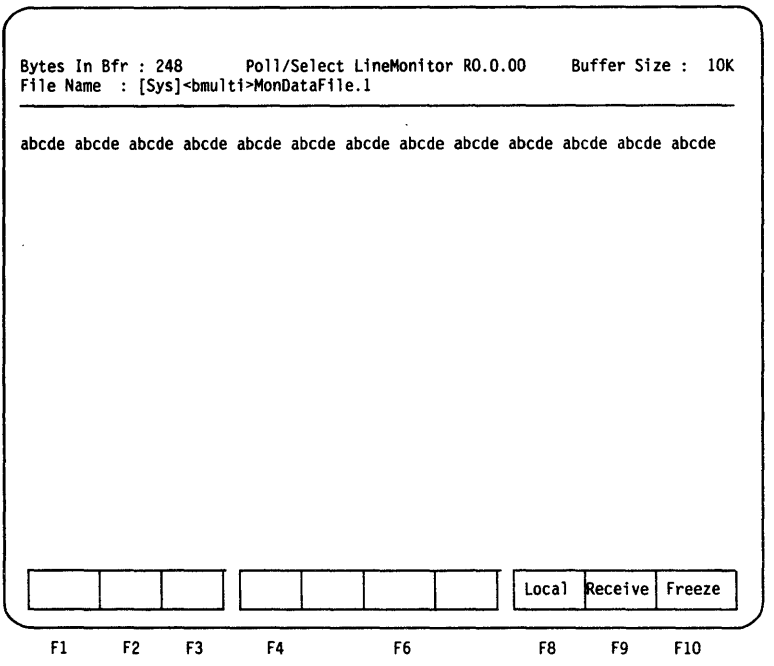
Buffer size can range from 1 to 60 K. To set the buffer size:

1 In local state, press BfrSize. The screen clears and displays the current buffer size.
2 Type the number of kilobytes to be allocated to the buffer.
3 Press GO.

Viewing Data from BMULTI

To monitor BMULTI, press RECEIVE. The line monitor is then logged on to BMULTI. As the screen fills with data, the newest data overwrites the oldest. A line is cleared ahead of the new data to indicate where new data begins. Data is sent to a ring buffer as described previously. Figure 5-5 illustrates a line monitor screen receiving test pattern "abcde."

Figure 5-5 **Sample Receive Session**



"Freezing" when Receiving Data

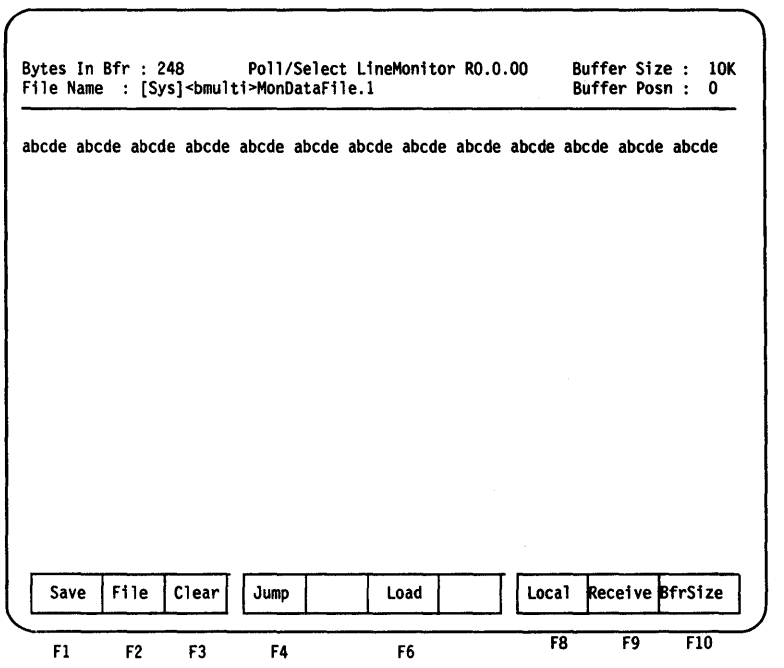
You can examine data more closely by pressing FREEZE or any unassigned key. This stops the flow of data to your screen. Freeze is a toggle; press it again to update the screen.

Although the screen is frozen, data continues to be sent to the ring buffer. When the screen is unfrozen, the screen displays the new information that was received while the screen was frozen.

Exiting Receive

To return to the local or offline state, press LOCAL or CANCEL. The window displays the last data received, though the menu options and captions change. Figure 5-6 shows the screen display from Figure 5-5 after exit from receive.

Figure 5-6 **Screen after Exit from Receive**



Loading a File to the Ring Buffer

You can load a file that was previously saved by the line monitor to the ring buffer. You can then use functions such as Jump to manipulate and/or examine the data.

Press LOAD. Enter the name of the file you want to examine. Press GO. Your window displays the file you loaded.

Use JUMP, and the SCROLL UP, NEXT PAGE, PREVIOUS PAGE, and SCROLL DOWN keys to manipulate the screen display. Use SAVE to save small portions of the file for closer study.

Save

All or part of a buffer can be saved using this function. From the local state:

1 With the data visible on your screen display, note the first offset and the number of bytes you want to save.

2 Press SAVE.

3 Enter a filename if you do not want to use the default.

4 Press GO. The screen displays:

Offset: (The offset at the beginning of the current display window.)

Number of Bytes: (The number of bytes following the current offset.)

5 Enter the location of the first offset you want to save. The number of bytes left in the buffer automatically appears. Enter the number of bytes you want to save.

6 Press GO.

Clearing the Buffer

From the local state, press CLEAR to erase the buffer.

"Jumping" and Scrolling Through Data

To "jump" to a specific offset in the buffer, use JUMP from the local state. The display begins with the offset you specify. You may then use the scrolling and paging keys to examine the information surrounding the chosen offset.

Figure 5-7 illustrates a file with an error at offset 97. Figure 5-8 illustrates this same file after the operator "jumped" to offset 89.

Figure 5-7 **Sample Buffer with Error "abQQe" at Offset 97**

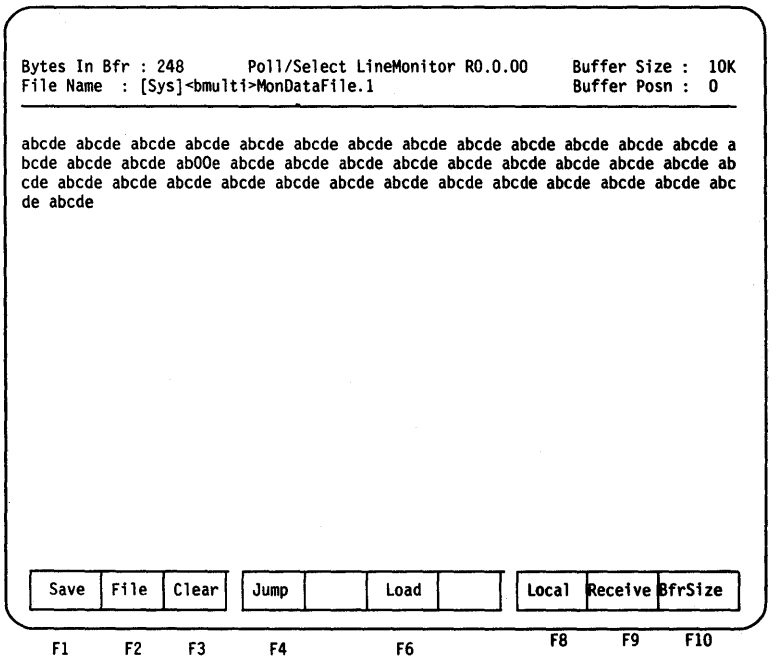
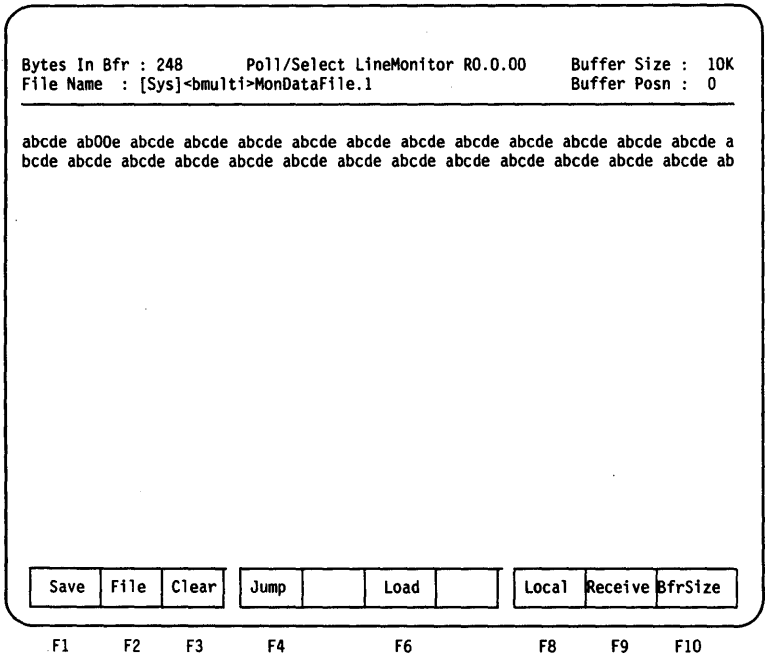


Figure 5-8 **Buffer shown in Figure 5-7 after JUMP to offset 89**



To jump:

1 Press JUMP.

2 Enter the offset to which you want to jump. The offset location is zero relative, the first offset in the buffer is number 0, the second offset is number 1, and so on.

3 Press GO.

Troubleshooting with the Line Monitor: A Sample Walkthrough

Suppose BMULTI seems to be working abnormally, and you suspect that this may have to do with the data being sent over the communications lines. These steps show one way to use the line monitor as a troubleshooting tool:

1 Enter the line monitor.
2 Change the filename (optional).
3 Set the buffer to an appropriate size.
4 Press RECEIVE.
5 Watch the data as it appears, freezing the display as needed. When you see information that appears to be unusual, return to local state.
6 Press JUMP and use the scrolling and paging keys to identify a sample of the file you want to save. Note the appropriate offset numbers.
7 Save this subset.
8 Later, you can load the saved file to a line monitor for further examination.

Basic BMULTI Concepts

This section contains recommendations for programming BMULTI and explains basic concepts. Sections 7 and 8 also acquaint you with Burroughs data communications.

Before you can use BMULTI, you must have an existing application or you must create one. To write the software, use the procedural interfaces described in Sections 9 and 10, and Appendix A. The procedural interfaces are object modules residing in BMULTI.lib. They format and issue requests to the BMULTI service. There are two procedural interfaces available:

1 High-level interface (HLI) suitable for use by application programmers
2 Enhanced low-level interface (LLI) suitable for experienced programmers who want a high degree of control over BMULTI's operation

Note: This release of BMULTI supports applications that were written using the Multiple-Task Interface (MTI) and Single-Task Interface (STI) from previous versions of BMULTI. You should use the HLI and LLI interfaces to create new applications or modify existing ones. Using these interfaces gives you the benefit of BMULTI 6.0 performance enhancements. For more information, see Sections 3 and 10, and Appendixes D, E, G, and H.

Recommended Interfaces

Situation	Language	Interface
New batch application	Pascal, COBOL, BASIC	HLI
New interactive application	Pascal	LLI
Enhancing older application	Any	HLI/LLI

This version of BMULTI includes support for extended character sets. This support is incorporated into BMULTI so that this feature need not be included in each individual application; it is available in one place for all applications running with BMULTI.

Linking Applications with BMULTI.lib

After writing and compiling your program, you must link it with BMULTI.lib. If you are using BASIC Interpreter or COBOL applications, link the interpreter with BMULTI.lib. See Appendix D for more information about configuring languages; Appendix A for sample programs.

Table 6-1 provides information about BMULTI.lib modules. See Table 1-1 for information about memory requirements for each module.

Table 6-1 BMULTI.lib Modules and Procedure Calls

Interface Used	BMULTI.Lib Module	Procedures Served
Multitasking Low-Level Interface	BMX1	BmCommand BmOpen BmReport BmReportWait BmOpenII
	BMX2	BmIdentify BmQuery BmStatus
Multitasking High-Level Interface	HLI6	CloseBMULTI OpenBMULTI ReadBMULTI ResetBMULTI SetOptionBMULTI

Note: The object module HLI6 uses the module BMX1.

Commands, Reports, and Requests

A BMULTI application instructs BMULTI through commands. BMULTI informs the application of events on the line by means of reports. Commands from an application may be accepted or denied by BMULTI depending on the state of the application's address. An address may move from one state to another in response to a command or an event on the line. Figure 7-1 shows BMULTI states.

The application issues commands and obtains reports by using procedural calls. The procedural calls issue operating system primitives called requests. These requests may either suspend themselves until a reply is available (wait), or continue to process, checking periodically to see if a reply is available (Check).

Virtual Addresses (VADs)

Virtual addresses (VADs) are used to avoid excessive line timeouts and to improve the performance of terminals on the same line. A VAD acts as if it were an active address in the IDLE state. The interface BMOpenII can be used to open an unused, previously defined VAD.

BMULTI searches for virtual addresses after searching for active addresses. If a poll is received for a virtual address, BMULTI always replies with EOT. If a select is received for a VAD, BMULTI always replies with NAK. BMULTI will always EOT polls and NAK selects for each inactive virtual address (i.e., no application is using that address).

See Appendix H for information about VADs and the host.

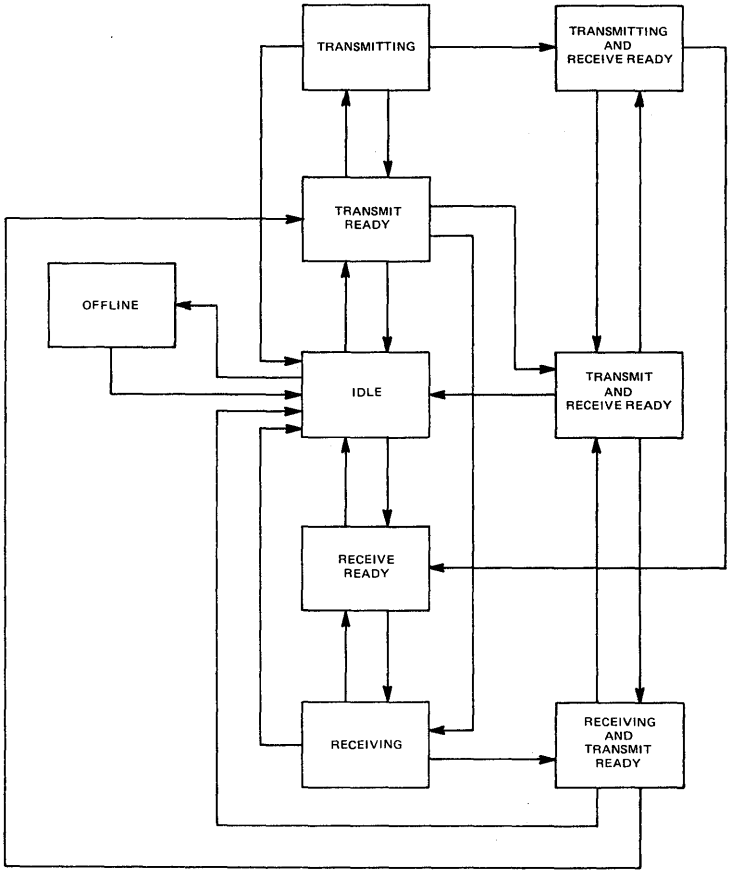
BMULTI State Machine

The current state of BMULTI affects the action it takes in response to a command from an application; therefore, it can be thought of as a state machine. The action depends on previous commands and events on the data communications line (which are passed to the application as reports). BMULTI may be in any one of nine states:

- Idle
- Offline
- Transmit ready
- Transmitting
- Transmitting and receive ready
- Transmit and receive ready
- Receiving
- Receive ready
- Receiving and transmit ready

Figure 7-1 shows the relationships between the states. BMULTI runs a parallel state machine for each address.

Figure 7-1 **BMULTI States**



The concepts used to explain the BMULTI states are:

Command Accepted and Command Denied: Accepted indicates that BMULTI returns a Command Accepted code and performs the requested action. Denied means that it returns a Command Denied code and does not perform the requested action.

Fast Ready Flag: The Set Fast Ready and Reset Fast Ready commands are accepted in all states except offline. These two commands alter the setting of an internal flag that BMULTI uses to determine the appropriate response to fast, group, and broadcast selects.

End Session Command: The End Session command is accepted from two states, offline and idle. The End Session command causes BMULTI to remove the associated address from the table of currently active addresses.

Reserving a Station Address

A configure command (such as BmOpenII) must be made first to reserve a station address. Until the application system issues such a call, any attempts by the host computer to select or poll the station will be ignored by BMULTI (except if it is a VAD). BMULTI continues to ignore any such attempts until an online command is issued. When a configure call is accepted by BMULTI, the application system is in the offline state.

Online, Offline, and Idle

To move an application from the offline state to the idle state, use an Online command. When idle, BMULTI replies to selects and polls addressed to the application station. BMULTI NAKs selects, and sends EOT in reply to polls. To return BMULTI to the offline state, use the Offline command.

Virtual addresses (VADs) always behave as if in the idle state.

Transmitting

Transmit Ready, when accepted, moves the application to the transmit ready state. In this state BMULTI looks for the next poll to the application's address. On seeing such a poll BMULTI issues a Ready for Transmit Buffer report. When returned, the application immediately issues a Transfer Transmit Buffer command to instruct BMULTI to obtain and transmit the buffer. When BMULTI sees an ACK from the host, it returns a Transmit Done report. If the application sees a Transmit Error report, it should look for another Ready for Transmit Buffer report and be prepared to issue another Transfer Transmit Buffer command.

Receiving

To move the application to the receive ready state, use the command Receive. In the receive ready state, BMULTI ACKs any subsequent selects and receives the transmitted message. If BMULTI does not detect an error in the message, it issues a Receive Done report. The application program should immediately issue a Transfer Receive Buffer command to retrieve the buffer because BMULTI keeps the buffer for a limited period of time. If the command is accepted, an ETX occurs after the last text character in the buffer. BMULTI issues the reports Receive Error, Duplicate Sequence Number, and Sequence Number Error (instead of Receive Done) when it receives a message in which it detects an error.

Idle and Abort

After a Receive or Transmit command is issued, you can use the Idle command to return the application to the idle state. However, if a select has already been ACKed or a transmission begun in response to a poll, use Abort to cause BMULTI to abandon attempts to receive or to transmit a message.

For example, if a message sent to the workstation contains embedded ETXs, BMULTI will not ACK it. BmReport will return a value of 7. After several successive returns, the application should issue an Abort command and either warn the operator or end the session.

Receiving Fast, Group, and Broadcast Selects

If the Set Fast Ready command is not issued, BMULTI NAKs any fast, group, or broadcast selects addressed to the application's station except when receive ready. If the Set Fast Ready command is issued, BMULTI ACKs and receives any such selects, even if it is not receive ready. (During the reception of a group or broadcast select, BmReport returns a value of 2 in order to distinguish between messages designated for that station in particular and messages designated for many stations.) After a Set Fast Select command has been issued, a Reset Fast Select command may be issued to prevent reception of fast, group, or broadcast select.

Terminating a Communication Session

The application terminates a communication session by issuing an End Session command. This command is accepted from only the idle and offline states.

Report Queue

BMULTI maintains a 10-deep queue of reports for each active address. Report codes Receiving, Receiving Group or Broadcast Select, Select Denied, and Receive Error are added to the report queue only if they are not already in the queue. Keep the report queue as shallow as possible by frequently reading it; if the queue is too deep, reports returned by BMULTI may be obsolete.

BMULTI States

Note: In the following discussion, the commands Set Fast Ready, Reset Fast Ready, and End Session are not listed. The preceding discussion applies.

Offline

Offline is the initial state of any address when BMULTI accepts a Configure command. In this state BMULTI ignores all control sequences for the assigned address. Only the following command is accepted. All others, except End Session, are denied.

Command	State Change
Online	Idle

Idle

In idle state, BMULTI is not ready to receive any data but is responsible for responding for the address. However, fast, group, and broadcast selects are accepted if the fast ready flag is on.

In this state, the following commands are accepted:

Commands	State Change
Offline	Offline
Idle	None
Receive	Receive ready
Transmit	Transmit ready
Transmit receive buffer	None
Abort	None

Inputs from the communications channel:

Poll of configured address:

State change:	None
Report:	None
Action:	Transmit an EOT

Group poll of installed group poll address:

State change:	None
Report:	None
Action:	If downstream RTS is FALSE and none of the other addresses assigned to this cluster are in transmit ready, transmit an EOT; otherwise no response.

Select of configured address:

State change:	None
Report:	Select denied
Action:	Transmit a NAK

Fast Select of configured address with fast ready flag set:

State change:	Receiving
Report:	Receiving
Action:	Wait for SOH

Fast Select of configured address with fast ready flag reset:

State change:	None
Report:	Select denied
Action:	Wait for the ETX and transmit a NAK

Broadcast select of configured address or group select of configured address with installed group select character, with fast ready flag set:

State change:	Receiving
Report:	Receiving group or broadcast select
Action:	Wait for SOH

Broadcast select of configured address or Group select of configured address with installed group select character, and fast ready flag reset:

State change:	None
Report:	Select denied
Action:	Wait for the ETX and transmit a NAK

Broadcast select of unconfigured address or group select of unconfigured address (any group select character), with fast ready flag reset:

State change:	None
Report:	None

Broadcast select of unconfigured address or group select of unconfigured address (any group select character), with fast ready flag set:

State change:	Receiving
Report:	Receiving group or broadcast select

Transmit Ready

In Transmit ready state, BMULTI is ready to transmit and is not ready to receive any data. The following table shows the acceptable commands and the state change they produce:

Command	State Change
Idle	Idle
Receive	Transmit and receive ready
Abort	Idle

Inputs from the communications channel:

Poll of configured address:

State change:	Transmitting
Report:	Ready for transmit buffer
Action:	Transmit message

Group poll of installed group poll address:

State change:	Transmitting
Report:	Ready for Transmit Buffer
Action:	Block downstream RTS and CTS; transmit message for each online application that is transmit ready, one by one.

Select of configured address:

State change:	None
Report:	Select denied
Action:	Transmit a NAK

Fast select of configured address with fast ready flag set:

State change:	Receiving and transmit ready
Report:	Receiving
Action:	Wait for SOH

Fast select of configured address with fast ready flag reset:

State change:	None
Report:	Select denied
Action:	Wait for ETX and Transmit a NAK

Broadcast select of configured address or group select of configured address with installed group select character, with fast ready set:

State change:	Receiving and transmit ready
Report:	Receiving group or broadcast select
Action:	Wait for SOH

Broadcast select of configured address or group select of configured address with installed group select character, with fast ready reset:

State change:	None
Report:	None
Action:	Wait for ETX and transmit a NAK

Broadcast select of unconfigured address or group select of unconfigured address with installed group select character, with fast ready set:

State change:	Receiving
Report:	Receiving group or broadcast Select

Transmitting

In transmitting state BMULTI has recognized a poll or group poll and is ready to transmit data on the communications channel. The following commands are accepted:

Idle (if CTS is not on):

State change:	Idle
Action:	Turn off RTS

Transfer transmit buffer:

State change:	None

Receive:

State change:	Transmitting and receive ready

Abort:

State change:	None
Action:	Set fast ready to FALSE, turn off RTS

Inputs from the communications channel:

EOT:

State change:	Transmit Ready
Report:	None
Action:	Unblock downstream RTS and CTS

ACK:

State change:	Idle
Report:	Transmit Done
Action:	If no more applications are Transmit ready (for a group poll), unblock downstream RTS and CTS. If downstream RTS is FALSE, transmit EOT. For specific poll, transmit EOT.

NAK:

State change:	None
Report:	None
Action:	Retransmit the data according to the protocol

RVI:

If specific poll:

State change:	Idle
Report:	Transmit Done
Action:	Transmit an EOT

If station transmitted last in reply to a group poll:

State change:	Idle
Report:	Transmit Done
Action:	Unblock downstream RTS and CTS. If downstream RTS is FALSE, transmit an EOT; otherwise no response.

If station is waiting to be unblocked and has not had an opportunity to reply to the group poll:

State change:	Transmit ready
Report:	Transmit Error
Action:	Unblock downstream RTS and CTS. If downstream RTS is FALSE, transmit EOT; otherwise no response.

Transmitting and Receive Ready State

In Transmitting and Receive ready state BMULTI has recognized a poll or group poll and is ready to transmit data on the communication channel. The addressed workstation is also ready to accept data. When the transmission is complete, the workstation will be in the Receive ready state.

The following commands are accepted:

Idle (if CTS is not on):

State change:	Idle
Action:	Turn off RTS

Abort:

State change:	Idle
Action:	Set Fast ready to FALSE. Turn off RTS

Inputs from the communications channel:

EOT:

State change:	Transmit and Receive ready
Report:	None
Action:	Unblock downstream RTS and CTS

ACK (for cluster stations individually in case of group poll):

State change:	Receive ready
Report:	Transmit Done

NAK:

State change:	None
Report:	None
Action:	Retransmit the data

RVI:

If specific poll:

State change:	Idle
Action:	Transmit an EOT

If station transmitted last in reply to a group poll:

State change:	Receive ready
Report:	Transmit Done
Action:	Unblock downstream RTS and CTS. If downstream RTS is FALSE, transmit an EOT; otherwise no response.

If station is waiting to be unblocked and has not had an opportunity to reply to the group poll:

State change:	Transmit ready and Receive ready
Report:	Transmit Error
Action:	Unblock downstream RTS and CTS. If downstream RTS is FALSE, transmit an EOT; otherwise no response.

Transmit and Receive Ready State

In Transmit and Receive ready state the protocol handler is ready to transmit and receive data. The following commands are accepted:

Idle:

State change:	Idle

Abort:

State change:	Idle

Inputs from the communications channel:

Poll of configured address:

State change:	Transmitting and Receive ready
Report:	Ready for Transmit Buffer
Action:	Block downstream RTS and transmit data

Group Poll of installed group poll address:

State change:	Transmitting and Receive ready
Report:	Ready for Transmit Buffer
Action:	Block downstream RTS and CTS

Select of configured address:

State change:	Receiving and Transmit ready
Report:	Receiving

Fast Select of configured address:

State change:	Receiving and Transmit ready
Report:	Receiving
Action:	Wait for SOH

Broadcast Select:

State change:	Receiving and Transmit ready
Report:	Receiving
Action:	Wait for SOH

Receiving State

In Receiving state the protocol handler is receiving a block of data. The command that is accepted is:

Abort:

State change:	Idle
Action:	Set Fast ready to FALSE
	Turn off RTS

Transmit:

State change:	Receiving and Transmit Ready

Inputs from the communications channel:

EOT:

State change:	Receive ready
Report:	None

ETX (and Block Check Character) (no Parity or BCC error):

State change:	Idle
Report:	Receive Done
Action:	Transmit and ACK if select was on my address

ETX (and Block Check Character) (Parity or BCC error):

State change:	None
Report:	Receive Error
Action:	Send NAK if select was on my address and wait for SOH or EOT

Receive Ready State

In this Receive Ready state BMULTI is ready to receive data. Commands accepted are:

Idle:

State change:	Idle

Transmit:

State change:	Transmit and Receive Ready

Abort:

State change:	Idle

Inputs from the communications channel:

Poll of configured address:

State change:	None
Report:	None
Action:	If downstream RTS is FALSE, transmit an EOT

Group Poll of configured group poll address:

State change:	None
Report:	None
Action:	If downstream RTS is FALSE, transmit an EOT.

Select of configured address:

State Change:	Receiving
Report:	Receiving

Fast Select of configured address:

State change:	Receiving
Report:	Receiving
Action:	Wait for SOH

Broadcast Select:

State change:	Receiving
Report:	Receiving
Action:	Wait for SOH

Group Select of installed group select character:

State change:	Receiving
Report:	Receiving
Action:	Wait for SOH

Receiving and Transmit Ready State

In Receiving and Transmit ready state BMULTI is receiving a block of data and is also ready to transmit on the next poll with this workstation's address. Commands accepted are:

Idle:

State change:	Idle

Abort:

State change:	Idle
Action:	Set Fast ready to FALSE

Inputs from the communications channel:

EOT:

State change:	Transmit and Receive ready
Report:	Receive Error

ETX (and Block Check Character) (No parity or BCC errors):

State change:	Transmit ready
Report:	Receive Done
Action:	Transmit an ACK if select was on my address.

ETX (and Block Check Character) (Parity or BCC errors):

State change:	None
Report:	Receive Error
Action:	Send NAK if select was on my address and wait for SOH or EOT.

Protocol Description

This section describes Burroughs multipoint terminal protocol. In this discussion, BTOS workstations running BMULTI applications are considered to be terminals.

Figures 8-1 through 8-7, at the end of this section, include all protocol options implemented in BMULTI. Most users need to know only a few of these.

The glossary contains more than twenty definitions that specifically apply to this section, including control character definitions.

Asynchronous Data Communication

In asynchronous data communication, each transmitted character uses ten nominally equal time intervals. The time intervals represent a start bit, eight bits of information, and a stop bit. Of the eight information bits, seven represent an ASCII character; the eighth is a parity bit selected to make the number of 1, or marking bits of the 8 bit group, even.

Synchronous Data Communication

In synchronous data communication, each transmitted character uses eight nominally equal time intervals, representing eight bits of information. The first seven bits represent the seven bit character code, transmitted with the least significant bit first. The eighth bit is to be a parity bit selected to make the number of 1, or marking bits of the eight bit group odd. The next transmission character follows immediately with no intercharacter interval.

Data Accountability

In transferring data from one point to another, proper accountability for each message is required under certain conditions. For example, in the handling of financial transactions, such as electronic transfer of funds, it is imperative that messages are not lost or duplicated. Where loss of a message or duplication is not important, transmission numbering may not be necessary. If each message sent has a transmission number serially assigned to it, the receiver can check that:

- Each message sent is received.
- Messages are received in the order they are sent.
- A message is not a retransmission of a previously transmitted message; therefore, it is not handled twice.

The message numbers sent from the host do not need to have a relationship to those sent from the other terminal and vice versa. Data transmission is not a balanced function; that is, one message sent does not always result in one reply.

Alternating Transmission Numbering

The minimum level of message numbering is a single character that alternates between an even and odd state. This system cannot distinguish between an error caused by message loss and an error caused by message duplication, though normal protocol procedures should prevent message loss. BMULTI allows for alternating a 0 and 1, or for alternating an @ and an A.

Sequential Transmission Numbering

Sequential transmission numbering provides more positive indication of message loss or duplication than the odd/even method. BMULTI allows one-, two-, or three-digit transmission numbers starting at 0 (00 or 000) and cycling through 9 (99 or 999).

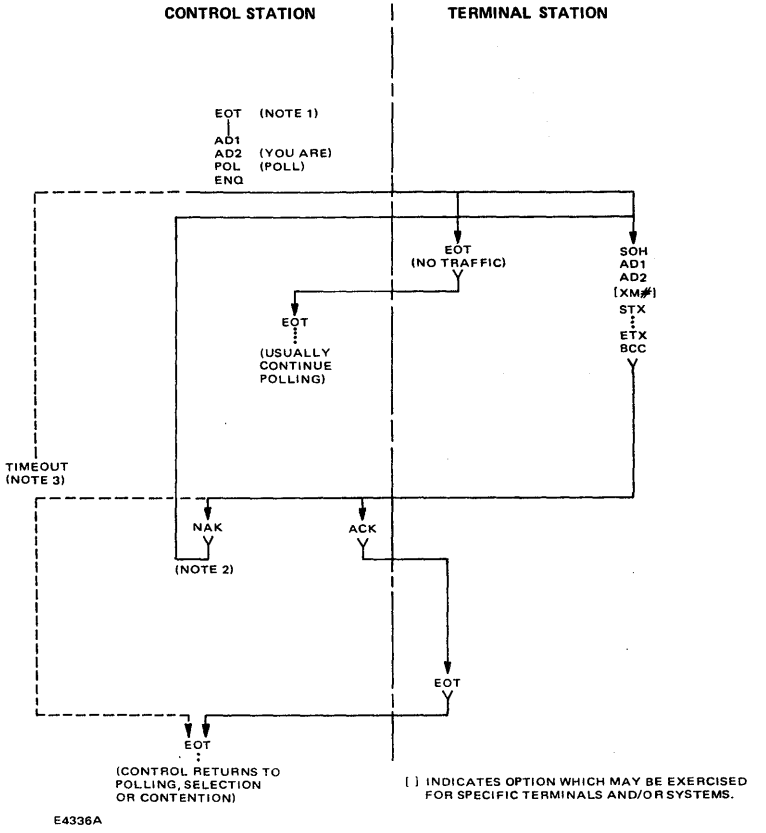
No Response Timeout

The timing starts after transmission of a character signifying reversal of transmission direction. The time ranges from one to three seconds. If the first character of a terminal transmission is not received, or if the character received is not valid in its time, the controller or terminal repeats its transmission 'n' times, in which 'n' is greater than or equal to 0; then, if the same condition exists, it interrupts and enters the necessary error recovery procedures. If the reversal is a result of an ACK or NAK, no repeat of the ACK or NAK is sent; however, EOT is sent to return to the control state.

Idle Line

The timing starts on receipt of each character other than a character signifying reversal of direction of transmission. Time ranges from one to three seconds. If the next character is not received in this time, the central processor interrupts and enters the necessary error recovery procedures.

Figure 8-1 **Specific Polling**



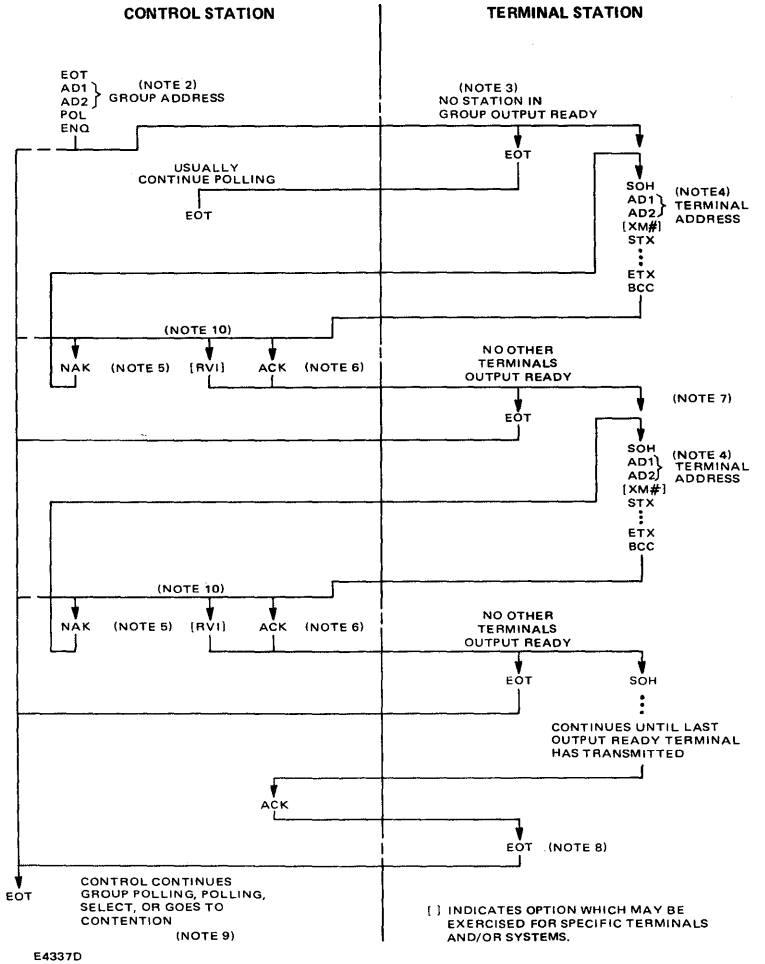
Notes to Figure 8-1

1 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the polling sequence may follow immediately.

2 If the control station receives a message for which character parity or block check test fails, NAK is transmitted, calling for a repeat of the transmission. This may be repeated n times (to be defined by the control station programmer), at which time, if the test fails, an error is recorded at the control station and EOT is transmitted, terminating the sequence. The terminal transmits the same message when next polled.

3 If the terminal does not receive ACK, NAK, or EOT, it may retain its message and remain quiet. The control station then times out and transmits EOT, terminating the sequence. In this case, the message is retransmitted when next polled.

Figure 8-2 **Group Polling**



Notes to Figure 8-2

1 This procedure is used to reduce the overhead in a network of terminals in which several terminals are located at one location on a common communication line. The receipt of one group poll results in one response for the group, if no terminals are output ready. Thus, the control station can pass to the next group. In periods of low activity, the control station has the ability to go through the polling list, determining the output status of all terminals with but one poll to each location, not each terminal. Also, if multiple terminals are output ready at a location, they are allowed to transmit, in sequence, in response to one poll. Selecting, broadcast, select, fast select, and so on are not affected by this group polling procedure.

2 In this procedure the polling sequence follows the same format as a normal poll and uses the normal poll character. Group polling is controlled by address only. Terminals at a common location that are to be a part of a group are so identified by making their group poll addresses all the same.

3 When the poll is received by the group addressed, the output ready terminals respond in the normal manner.

4 Each message sent in response to a group poll contains the address of the individual terminal that is responding.

5 If the control station detects an error in the message received in response to a group poll, normal polling error recovery is used.

6 The control station must, under this procedure, be sure when it replies ACK to a message that buffer space exists or is to be available for the next message that could result from another output ready terminal.

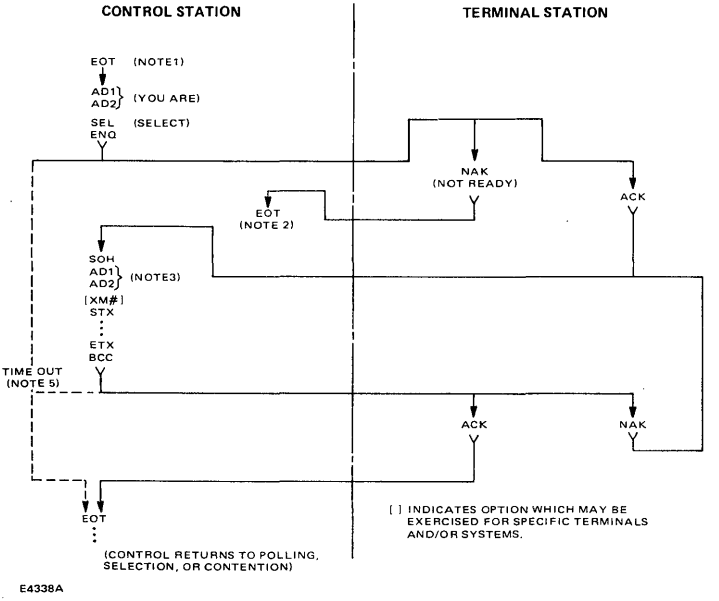
7 As soon as ACK is received from the control station, the next output ready terminal transmits.

8 When an ACK is received from the control station and no terminals remain output ready, the last terminal to transmit is responsible to transmit the final EOT.

9 The same error recovery procedure outlined in figure 8-1 is used with this procedure.

10 Reverse interrupt (RVI) may be used by the control station only after receipt of a valid message that would result in a positive acknowledgment. In place of sending ACK, the control station sends RVI (DLE<).

Figure 8-3 **Selection**



Notes to Figure 8-3

1 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the selection sequence may follow immediately.

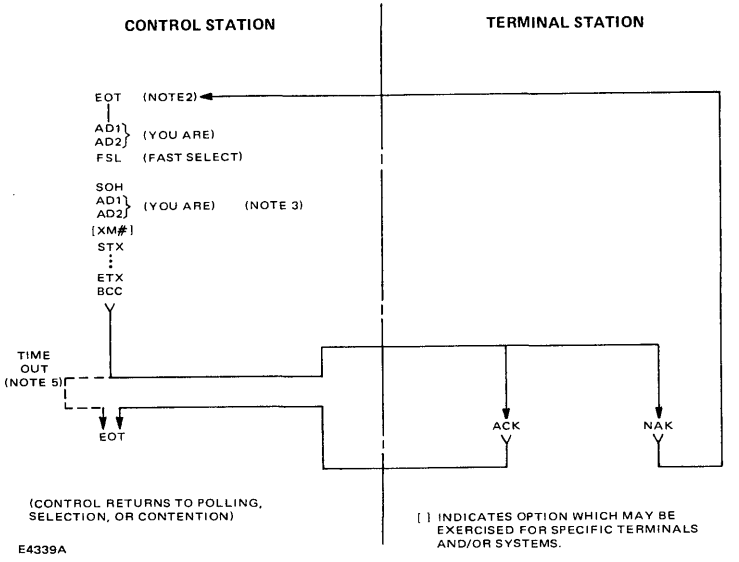
2 If the terminal is not ready to receive, as indicated by transmission of NAK, the control station normally retries the selection at the proper sequence of that terminal.

3 The identification characters in a transmission represent the terminal address for selection verification purposes. If the terminal fails to verify the address, it ignores the message.

4 If character parity or block check are not validated by the terminal, it sends NAK. In this case the control station retransmits the message n times (n may be equal to zero). If the terminal still does not acknowledge the message, the control station terminates the sequence with EOT, after recording the error. The control station retains the message for transmission on the next selection sequence to this terminal.

5 If the control station does not receive a response (ACK or NAK) to its message, it may time out and retransmit the message n times (n may equal zero). If still no response is received, the control station terminates the sequence with EOT, after recording the error. The control station retains the message for transmission on the next selection sequence to this terminal.

Figure 8-4 **Fast Select**



Notes to Figure 8-4

1 Fast selection is used when the control station wants to send a message to a terminal without first testing to make sure that the terminal is ready to receive. In this case, the selection and the message are transmitted together. The ACK response from the terminal applies both to the select and to a successful message transfer. A NAK response may indicate either that the terminal is not ready to receive or that the parity or block check in the message is invalid.

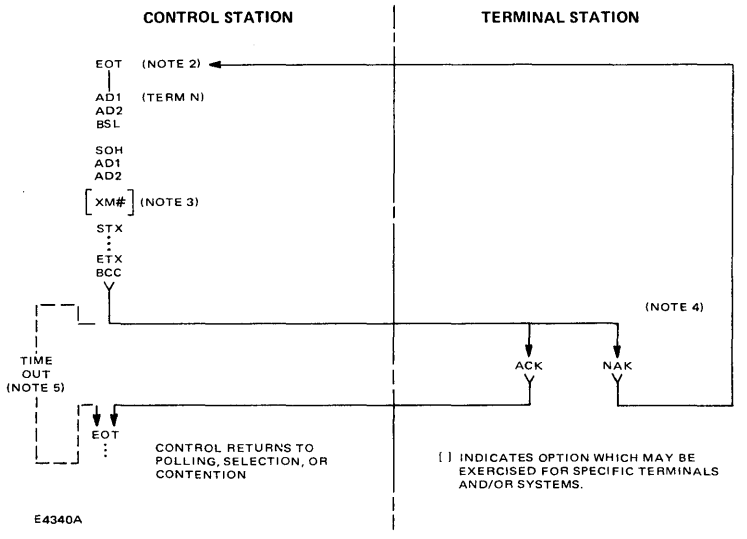
2 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the fast selection sequence may follow immediately.

3 The identification characters in a transmission from the control station also represent the terminal address YOU ARE for selection verification purposes. If either pair of addresses fail to verify, the terminal ignores the message.

4 If character parity or block check are not validated by the terminal selected, or if the terminal selected is not ready to receive the message, it responds NAK. In this case, the control station resends the fast select transmission n times (n may equal zero). If the terminal still does not accept the message, the control station terminates the sequence and retains the message for transmission on the next selection sequence to this terminal.

5 If the control station does not receive a response (ACK or NAK) to its transmission, it times out and terminates the sequence. The control station retains the message for transmission on the next normal selection for this terminal.

Figure 8-5 **Broadcast Select**



Notes to Figure 8-5

1 Broadcast select is a fast selection of all terminals. AD1-AD2 is selected to represent the terminal that acknowledges receipt of the message.

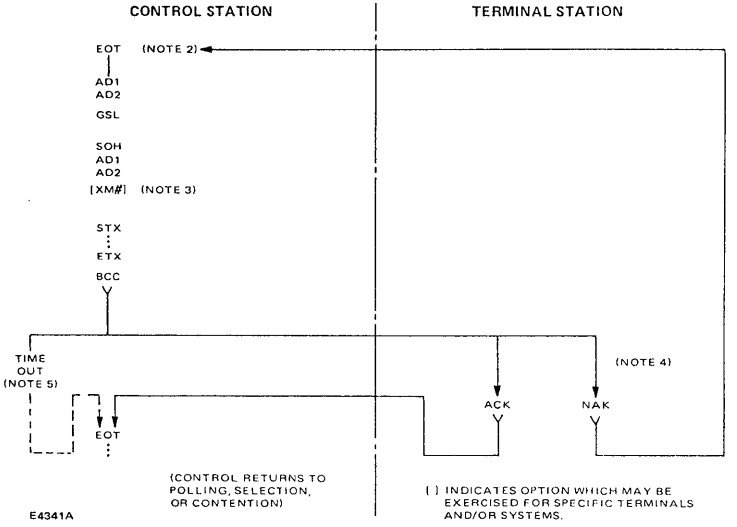
2 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the broadcast sequence may follow immediately.

3 Special sequences of numbers must be maintained if transmissions are numbered in a system in which broadcast is employed.

4 If the acknowledging terminal does not receive a valid message (e.g., there is a character parity or block check error) or is not receive ready, it transmits NAK. The control station has the option of repeating the entire broadcast.

5 If the control station does not receive a response (ACK or NAK) to its broadcast, it may time out and rebroadcast the message n times (n may equal zero). If no response is received, the control station terminates the broadcast mode with EOT after recording the error.

Figure 8-6 **Group Select**



Notes to Figure 8-6

1 Group selection is a fast selection of a group of terminals. Each terminal may have a group select character for which it accepts a message. AD1-AD2 is selected to represent the address of the terminal that acknowledges receipt of the message.

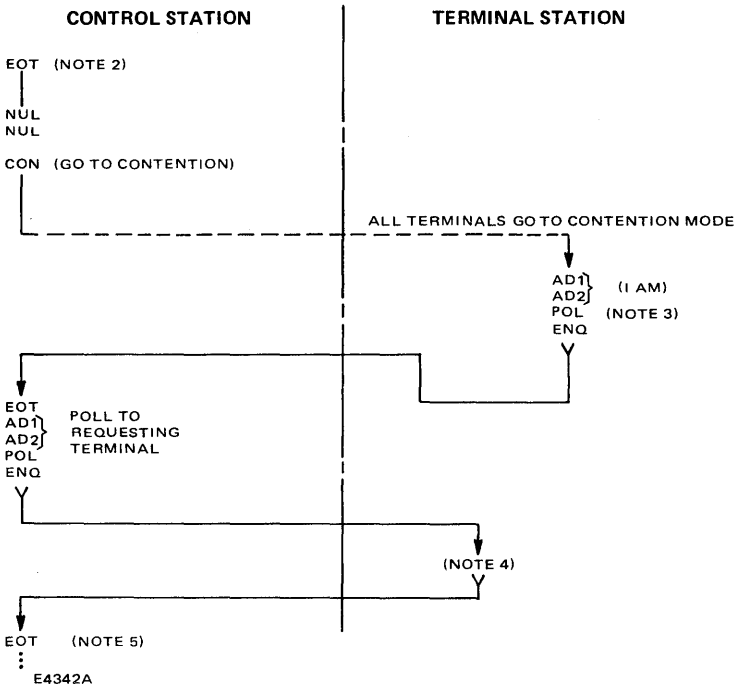
2 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the group selection may follow immediately.

3 Special sequences of numbers must be maintained if transmissions are numbered in a system in which group selection is employed.

4 If the acknowledging terminal does not receive a valid message (e.g., there is character parity or block check error) or it is not receive ready, it transmits NAK. The control station has the option of repeating the entire group selection.

5 If the control station does not receive a response (ACK or NAK) to the group selection, it may time out and reselect the group n times (n may equal zero). If no response is received, the control station terminates the group select mode with EOT after recording the error.

Figure 8-7 **Multipoint Contention Mode**



Notes to Figure 8-7

1 In times of low activity, it may be desirable to terminate polling and to place all or part of the system in the contention mode. This is done by transmission of EOT NUL NUL CON which causes the terminals to remain quiet until they have something to transmit.

2 This EOT must come from the control station and may have been the termination of a previous transmission sequence. To minimize the effect of noise, the go to contention sequence may follow immediately.

3 A terminal may wake up the polling activity by transmitting AD1 AD2 POL ENQ. This causes the control station to poll that terminal. If two terminals attempt to transmit at the same time, the garbled message initiates general polling by the control station.

4 The terminal proceeds with normal message transfer as in response to a poll (see Figure 8-1).

5 Following normal message receipt verification procedures, as in Figure 8-1, the control station may continue polling or instruct all terminals to go to contention.

High-Level Procedural Interface

Of the four interface levels provided by BMULTI, the high-level interface (HLI) is the easiest to use. (The STI and MTI interfaces are supported but not recommended for new program development.) We recommend that you use the HLI to create COBOL or BASIC applications. See Appendix A for sample programs.

These procedures are described in this section:

- CloseBMULTI
- OpenBMULTI
- ReadBMULTI
- ResetBMULTI
- SetOptionBMULTI
- WriteBMULTI
- SetXlatModeBMULTI
- SelectBMULTI

The High-Level and Low-Level Interfaces support request routing over B-NET. A properly written application linked with either interface can access BMULTI running on a remote B-NET node. Read the *BTOS Systems B-NET Administrator's Guide* for more information about B-NET.

The application status block (ASB) is used by the HLI to communicate the status of the reads and writes issued by an application. This block's format is shown in Table 9-1.

Note: Programmers who are using Pascal should use explicit offsets when declaring this structure.

Table 9-1 **Format of the Application Status Block**

Field	Size bytes	Appli-cation Usage	Comment
RcvStatus	1	R	Used by HLI process to indicate Read status.
RcvErc	2	R	Internal error seen by the HLI process for receive.
fSelDen	1	R/W 0	Set by HLI process to TRUE if line is selecting this address. An application should reset this flag to 0 after processing.
Xmtstatus	1	R	Used by HLI process to indicate write status.
Xmterc	2	R	Internal error seen by the HLI process for XMT.
Option	1	R	Station option byte (used by SetOptionBMULTI).
fFMess	1	R/W 0	Set to TRUE if a fast select message has been received by the HLI process for this address. An application should reset to this flag to 0 after processing.
pFMess	4	R	Address of fast message buffer (which is not in the application area). This pointer is returned by the HLI process after an OpenBMULTI.

CloseBMULTI

Description

CloseBMULTI frees the device address by issuing an end session command. This is normally the last step in a data comm session.

Procedural Interface

CloseBMULTI (Sh) : Erctype

Sh	byte (returned by OpenBMULTI).

OpenBMULTI

Description

OpenBMULTI opens a BMULTI station with the supplied device address and returns a station handle that should be used for all successive BMULTI calls. OpenBMULTI also creates the HLI process. The application must provide an ASB.

Procedural Interface

OpenBMULTI (devAdr, fSys, Priority, pSh, pASBlk) : Erctype

devAddr	A word containing two ASCII characters. This address must not duplicate that of any other stations on the same line.
fSys	A byte or Boolean. It should be set to TRUE if the application making the call is to be a system service.
Priority	A word, containing the priority with which the HLI process is to be created. (This is normally higher than 128, e.g., 127.)
Sh	A pointer to a byte.
pASBlk	A pointer to the application-supplied ASB. The format of the ASB is shown in Table 9-1.

ReadBMULTI

Description

Read BMULTI initiates a receive operation. It does not wait for the message to arrive. The application should sample the ASB Rcvstatus to determine when the receive is complete. When the receive is complete, the HLI process returns the received message in the receive buffer. The first two bytes contain the length of the received message.

You are responsible for creating a receive buffer of adequate size. If the buffer does overflow (i.e., RcvStatus field in ASM equals 14), you should issue a delay of approximately 1 second or issue a ResetBMULTI prior to issuing another ReadBMULTI.

Procedural Interface

ReadBMULTI (Sh, pBuf, sBuf) : Erctype

Sh	A byte (returned by OpenBMULTI).
pBuf	A pointer to an application-supplied buffer into which the received data is to be placed.
sBuf	A word containing the maximum number of bytes the application can receive.

Possible hexadecimal values of RcvStatus:

00	Initial state or after a reset
01	Busy (read initiated)
02	Read failure
10	Read complete (no errors)
11	Read complete (transmission number error)
12	Read complete (duplicate transmission number)
14	Read complete (truncated message)

ResetBMULTI

Description

Reset BMULTI restores the application status to idle and aborts an ongoing read or write. After a reset, the option byte is set to 0.

Procedural Interface

ResetBMULTI (Sh) : Erctype

Sh	A byte (returned by OpenBMULTI).

SetOptionBMULTI

Description

SetOptionBMULTI enables the application to select the BMULTI options: offline, online, fast ready, or not fast ready. Section 7 explains these options.

Procedural Interface

SetOptionBMULTI (Sh, Option) : Erctype

Sh	A byte (returned by OpenBMULTI).	
Option	A byte with values as follows:	
	Bit 0	1 = Offline 0 = Online
	Bit 1	1 = Fast Rdy 0 = Not Fast Rdy

The ASB option field is updated to reflect the option selected.

WriteBMULTI

Description

WriteBMULTI initiates a transmit operation, but does not wait for the message to be successfully transmitted. The application should sample the ASB Xmtstatus to determine when the transmission is complete.

Procedural Interface

WriteBMULTI (Sh, pBuf, sBuf) : Erctype

Sh	A byte (returned by OpenBMULTI).
pBuf	A pointer to an application-supplied buffer that contains the data BMULTI is to transmit.
sBuf	A word containing the number of bytes BMULTI is to transmit.

Possible hexadecimal values of XmtStatus:

00	Initial state or after a reset
01	Busy (write initiated)
02	Write failure
10	Write complete

SelectBMULTI

Description

The SelectBMULTI procedure is part of the High-Level Interface (HLI). It allows the user to use the copy of BMULTI that resides on another node to transfer files. If no remote node is named, the local copy of BMULTI is used. The SelectBMULTI call must be made immediately before each OpenBMULTI to be effective, because the node specification entered as part of the SelectBMULTI call is stored for use by the OpenBMULTI procedure. Since this specification is used by each OpenBMULTI call that follows, the SelectBMULTI procedure should be used before each OpenBMULTI procedure. The use of SelectBMULTI is optional. If it is not used, the node selection defaults to the local copy of BMULTI.

Procedural Interface

SelectBMULTI (pbNodeName, cbNodeName): Erctype;

pbNodeName/cbNodeName	**A B-NET node name string that defines the location of the BMULTI to be used.**

SelectBMULTI is an object module procedure.

SetXlatModeBMULTI

Description

SetXlatModeBMULTI is used to select the character translation algorithm to be used with extended character sets. This procedure works the same way as the BmSetXlatMode procedure included in the Low-Level Interface. The commands to open BMULTI (all interfaces) establish the default mode as 0 (no translation required). Therefore, the use of this procedule is optional . SetXlatModeBMULTI can be used to start, terminate, or change the existing character translation mode. Although the mode is individually selectable for each station, the translation table specified in the configuration file is common for all users of this copy of BMULTI.

To use this procedure, BMULTI must be off line, or else in the idle state and not fast-ready. If these conditions are not met, or if an invalid mode is specified, an error code 32806 is returned.

Procedural Interface

SetXlatModeBMULTI (Sh, iRcvMode, iXmitMode): Erctype;

Sh	The station handle returned by OpenBMULTI.
iRcvMode	Selects the character translation algorithm to be used for transmitting characters. This parameter provides the same functionality as the iRvcMode described for the BmSetXlatMode procedure.
iXmitMode	Selects the character translation algorithm to be used for transmitting characters. This parameter provides the same functionality as iRcvMode described for the BmSetXlatMode procedure.
SetXlatModeBMULTI	An object module procedure.

Low-Level Interface

This section describes the BMULTI low-level procedural interface (LLI). The LLI is useful for intricate applications that require extra control.

These procedures are described in this section:

- BmCommand
- BmOpenII
- BmReport
- BmReportTimeout
- BmReportWait
- BmGetStatus
- BmSetXlatMode

B-NET Support

The LLI supports request routing over B-NET. A properly written application linked with the LLI can access BMULTI running on a remote B-NET node. Read *BTOS Systems B-NET Administrator's Guide* for more information about B-NET.

BmCommand

Description

BmCommand passes commands from the application to BMULTI.

Unlike the previous version of BMULTI, BmCommand returns without a wait if the command was a Transfer Transmit Buffer. It returns ercOk (0). BmCommand checks whether the request block returned to the wait was the one made in the previous request, not the Transfer Transmit Buffer request. If it is the Transfer Transmit Buffer request, BmCommand does another wait.

Procedural Interface

BmCommand (Sh, Command, pBuff, sBuff) : Erctype

Sh	A byte (the value of which is returned by BmOpenII).
Command	A word containing a value from the following table.

Possible values of Command are:

1	Get receive buffer*
2	Send transmit buffer*
3	Offline
4	Online
5	Idle
6	Set fast ready
7	Set receive ready
8	Set transmit ready
9	End session
10	Abort
11	Reset fast ready
12	Transmit extended message* (Messages >2048)

* These commands require valid pBuff and sBuff.

pBuff

sBuff	describe the application's message buffer. These are normally dummy values except for the three buffer transfer commands (1, 2, and 12). For Get Received buffer, sBuff should be set to the maximum number of bytes that the application can accept. After BmCommand returns, the first two bytes starting at pBuff contain the number of bytes received. For Send Transmit buffer, sBuff should be set to the number of bytes the application wants to transmit.

BmOpenII

Description

BmOpenII is the first procedure called by BMULTI applications. With BmOpenII, you can request that an unused virtual address (VAD) be used instead of a specific address. (See Section 6 and Appendix H for more information about VADs.) It also provides support for B-NET (LANs) and multiple copies of BMULTI per cluster system. If you provide a node name, it is saved by the object module and used in the other requests. It is important to replace BmOpen with BmOpenII before linking older applications with 6.0 BMULTI. If this is not done, the application will not be able to use certain 6.0 features such as 50 ms host timeout.

Note: Applications that call BmOpenII with the "any open VAD" option must be introduced very carefully into existing installations, especially if they are being mixed with older applications that have fixed addresses. See Appendix F for more details.

Procedural Interface

BmOpenII (DevAdr, pShRet, fSys, pDevAdrRet, pbHostName, cbHostName, pbDescString, cbDescString) : Erctype

DevAdr	A word that indicates the device address. If this word contains OFFFFh, the first unused VAD is selected. If the word contains any other value, that value is used.
pShRet	A pointer to a byte allocated by the application, into which the BMULTI station handle is returned.
fSys	A byte or Boolean. This flag determines whether termination requests for this userNum are to be honored. Set it to TRUE if the application making the call is a system service.

pDevAdrRet	A pointer to a word allocated by the application. The value of the actual DevAdr used is returned here.
pbHostName, cbHostName	A B-NET node name, for example, ?AlphaCentauri? or ?Philadelphia?. This string must contain a node name at the beginning. If cbHostName is zero, the application is connected to the default route for BMULTI requests (BMULTI on the local node).
pbDescString, cbDescString	A string (maximum length of 8 bytes) that describes the application (for example, 'PPT', 'HostLink', 'Infoview', 'BTE', 'DTS'). If cbDescString is zero, the status monitor will not list the application name.

BmReport

Description

BmReport returns the status of the data comm subsystem to the application.

If erctype zero is returned, either there are no errors or no report is available. If a value other than zero is returned, the report is not valid.

Procedural Interface

BmReport (Sh, pReport) : Erctype

Sh	A byte (this value is returned by BmOpenII).
pReport	A pointer to a word into which the procedure is to return the report.

BmReport returns the following reports:

0	No report
1	Receiving
2	Receiving Group Or Broadcast Select
3	Receive Done
4	Ready for Transmit buffer
5	Transmit Done

6	Select Denied
7	Receive Error
8	Duplicate Transmission Number
9	Transmission Number Error
10	Transmit Error

BmReportTimeout

Description

BmReportTimeout waits for a report for a specified period of time. The report is valid only if the value zero is returned.

Procedural Interface

BmReportTimeout (Sh, pReport, timeout) : Erctype

Sh	A byte (this value is returned by BmOpenII).
pReport	A pointer to a word into which the procedure is to return the report.
timeout	A word that gives an interval (in tenths of a second) during which the procedure waits for a report.

BmReportTimeout returns the same reports as BmReport.

BmReportWait

Description

BmReportWait waits for a report indefinitely. The report is valid only if the procedure returns zero.

Procedural Interface

BmReportWait (Sh, pReport) : Erctype

Sh	A byte (the value of which is returned by BmOpenII).
pReport	A pointer to a word to which the procedure is to return the report.

BmReportWait returns the same reports as BmReport.

BmGetStatus

Description

BmGetStatus provides the caller with release-independent access to BMULTI information. It consolidates the necessary data returned by the BmQuery and BmIdentify routines of the 4.0 and 5.0 releases of BMULTI and presents this data in condensed, usable formats. BmQuery and BmIdentify are not supported in BMULTI 7.0 and higher.

Procedural Interface

BmGetStatus (pbNodeName, cbNodeName, iStation, nStation, iRecordLen, pbCommonRet, cbCommonRet, pbStatRecsRet, cbStatRecsRet): Erctype;

pbNodeName	B-Net node name where BMULTI is installed. If empty string is specified, default is local BMULTI or BMULTI of master for local station if no local BMULTI is installed.
cbNodeName	B-Net node name where BMULTI is installed. If empty string is specified, default is local BMULTI or BMULTI of master for local station if no local BMULTI is installed.
iStation	Index of the first station for which information is to be returned.
nStation	Number of station records required.
iRecordLen	Length of each data record to be returned. If the total length of all records to be returned (iRecordLen * nStation) is greater than the length of the specified buffer (cbStatRecsRet), an error erclnvalidRecordLength will be returned.
pbCommonRet	Description of a buffer where the general information of the same environment is to be returned.
cbCommonRet	Description of a buffer where the general information of the same environment is to be returned.

pbStatRecsRet	Description of a buffer where station status records are to be returned.
cbStatRecsRet	Description of a buffer where station status records are to be returned.

Common Information Buffer Format

Field	Offset	Bytes	Definition
fLineActivity	0	2	Counter incremented by all transmit and receive interrupts.
WsNumber	2	2	Workstation number.
Channel	4	2	Two characters specify which port the station is using.
NakCount	6	2	NAK counter for this copy of BMULTI. Maximum 65535.
nActive Stations	8	2	Number of active stations. Maximum 64.
Mode	10	1	*A* if asynchronous or *S* if synchronous.
BMULTIVersion	11	13	Installed BMULTI version format. String length field is 1 byte. The character string field is a maximum of 12 bytes. The actual character string can be a maximum of 11 characters. The string is terminated by a null, which is used as an end-of-string delimiter in some languages. The null character is not counted when determining string length.

Station Record Format

Field	Offset	Bytes	Definition
StationIndex	0	2	The index of the station record.
DeviceAddr	2	2	The BMULTI line address.
UserNum	4	2	Application user number as seen by BMULTI.
State	6	1	Current state of the station, where: 0 = offline 1 = idle 2 = receive ready 3 = receiving 4 = transmit ready 5 = transmit and receive ready 6 = receiving and xmit ready 7 = transmitting 8 = transmitting and receive ready
Rqs	7	1	Number of outstanding requests.
Xmit	8	12	A 4-record array of 3-byte text strings. The records contain the expected transmission numbers for transmit, receive, group select, and broadcast select.
TxMsg	20	2	Count of transmitted messages.
RxMsg	22	2	Count of received messages.
StartTime	24	4	Date/time when user first logs onto BMULTI.

ApplDesc	28	Up to 10	User application passes in the open command. *Format*: String length (1 byte) Character string (up to 9 bytes) The actual character string is only 8 characters long. The string is terminated with a null. The null is not included when giving the string length.
UserName	38	Up to 10	Application user name. String length of 1 byte and a character string of maximum 8 bytes followed by a null character. The null character is not counted when determining string length.

Description

BmSetXlatMode is used to select the character translation algorithm to be used with extended character sets. The commands to open BMULTI (all interfaces) establish the default mode as 0 (no translation required). Therefore, the BmSetXlatMode procedure is optional. It can be used to start, stop or change the existing character translation mode. Although the mode is individually selectable for each station, the translation table loaded for the configuration file is common for all users of this copy of BMULTI. Examples are found in Appendix J, translation tables in Appendix K.

To avoid errors, BMULTI must be offline or in the idle state and not fast-ready when using BmSetXlatMode.

Procedural Interface

BmSetXlatMode (Sh, iRcvMode, iXmitMode): Erctype;

Sh	The station handle returned by BmOpen or BmOpenII.
iRcvMode	The character translation algorithm to be used for received characters. Enter *0* for no translation (all characters pass through). Enter *1* if the ESC SO escape sequence indicates that a string of encoded extended characters follows. The ESC SI escape sequence indicates that a string of standard (not extended) characters follows. When this mode is selected, these escape sequences are removed from the data stream when received. Enter *2* if the ESC SO and the SO character alone each serve as indicators that a string of encoded extended characters follows. The ESC SI and SI character alone each indicate that a string of standard (not extended) characters follows. When this mode is selected, these sequences or characters are removed from the data stream when received. Enter *4* if there is to be no decoding of the data stream. All characters in the message received will be translated using the translation table. Enter *5* if the ESC SO escape sequence indicates that a string of encoded extended characters follows. The ESC SI escape sequence indicates that a string of standard (not extended) characters follows. When this mode is selected, these escape sequences are removed from the data stream when received. In addition, all characters received are translated after any necessary decoding is accomplished. Enter *6* if the ESC SO and the SO character alone each serve as indicators that a string of encoded extended characters follows. The ESC SI and SI character alone each indicate that a string of standard (not extended) characters follows. When this mode is selected, these sequences or characters are removed from the data stream when received. In addition, all characters received are translated after any necessary decoding occurs.

iXmitMode

Selects the character translation algorithm to be used for transmitting characters.

Enter *0* for no translation. All characters pass through. The high-order bit of the byte value of any extended character is stripped away in the transmission process.

Enter *1* to insert the sequence ESC SO into the data stream to prefix a string of encoded extended characters, and to insert the sequence ESC SI to prefix a string of standard characters.

Enter *2* to insert only the SO character into the data stream to prefix a string of encoded extended characters. Only the SI character will be inserted into the data stream to prefix a string of standard characters.

Enter *4* if there is to be no encoding of the data stream. All characters in the message will be translated prior to transmission. The high-order bit of the byte value of any extended character will be lost in the transmission process.

Enter *5* to insert the sequence ESC SO into the data stream to prefix a string of encoded extended characters, and to insert the sequence ESC SI to prefix a string of standard characters. All characters in the message will be translated before determining whether encoding is necessary.

Enter *6* to insert only the SO character into the data stream to prefix a string of encoded extended characters. Only the SI character will be inserted into the data stream to prefix a string of standard characters. All characters in the message will be translated before determining whether encoding is necessary.

Sample Programs

This appendix contains sample programs written in COBOL, BASIC, FORTRAN, and Pascal. The COBOL, BASIC, and FORTRAN programs use the high-level interface. There are two Pascal programs; the first uses the high-level interface and the second uses the low-level interface

Applications written in any language will not operate unless BMULTI.lib has been linked with the executable code file. Applications written in FORTRAN, BASIC, or COBOL will not operate unless the languages have been reconfigured. Appendix D contains details about language configuration.

COBOL Echo Program (Using High-Level Interface)

This program echoes any text received back to the host system.

```
000100 IDENTIFICATION DIVISION.
       PROGRAM-ID.  Echo.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  B20.
       OBJECT-COMPUTER.  B20.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       77  counter                 PIC 9(04) COMP.
       01  miscellaneous.
           03  device-address      PIC XX.
           03  ercdisplay          PIC 9(05).
           03  max-buffer-size     PIC 9(04) COMP VALUE 4096.
           03  online              PIC X VALUE X"00".
           03  fss                 PIC X.
           03  pri                 PIC 9(04) COMP.
           03  Sh                  PIC X VALUE X"00".
           03  buffer-size         PIC 9(04) COMP.
           03  buffer-size-a2      REDEFINES buffer-size PIC X OCCURS 2.
      *
       01  buffer-whole.
           03  buffer-size-a       PIC X  OCCURS 2.
           03  buffer              PIC X(4096).
           03  buffer-array        REDEFINES buffer.
               05  byte            PIC X  OCCURS 4096.
      *
       01  Error-code              PIC 9(04) COMP.
           88  Address-Is-Good     VALUE 0.
           88  No-Error            VALUE 0.
```

```
*
 01  asBlk.
     03  RcvStatus          PIC X.
         88  Idle           VALUE X"00".
         88  ReadBusy       VALUE X"01".
         88  ReadErr        VALUE X"02".
         88  ReadDone       VALUE X"10".
         88  SeqErr         VALUE X"11".
         88  DupSeq         VALUE X"12".
         88  TruncMsg       VALUE X"14".
     03  RcvErc             PIC 9(04) COMP.
     03  fSelDen            PIC X.
     03  XmtStatus          PIC X.
         88  Local          VALUE X"00".
         88  WriteBusy      VALUE X"01".
         88  WriteErr       VALUE X"02".
         88  WriteDone      VALUE X"10".
     03  XmtErc             PIC 9(04) COMP.
     03  Opt                PIC X.
     03  fMess              PIC X.
     03  pfMess             PIC X(04).
*
 PROCEDURE DIVISION.
 MAIN-LINE.
     PERFORM start-up.
     PERFORM driver THRU driver-x
         VARYING counter FROM 1 BY 1
             UNTIL counter IS EQUAL TO 50.
     PERFORM finish-up.
*
 START-UP.
     DISPLAY "BMULTI echo program" UPON CONSOLE.
     PERFORM Get-Address THRU Get-Address-X
         UNTIL Address-Is-Good.
     DISPLAY "Begin BMULTI" UPON CONSOLE.
     CALL "&SETOPTIONBMULTI" USING error-code, Sh, online.
     PERFORM error-check.
*
 DRIVER.
     MOVE SPACES TO buffer.
     CALL "&READBMULTI" USING error-code, Sh, buffer-whole,
                              max-buffer-size.
     PERFORM Error-Check.
     PERFORM Check-Read-Complete.
     MOVE buffer-size-a (1) TO buffer-size-a2 (2).
     MOVE buffer-size-a (2) TO buffer-size-a2 (1).
     MOVE buffer-size TO ercdisplay.
     DISPLAY "Message size is: ",ercdisplay UPON CONSOLE.
     DISPLAY "Counter is  ",counter UPON CONSOLE.
     CALL "&WRITEBMULTI" USING error-code, Sh, buffer,
                               buffer-size.
     PERFORM Error-Check.
     PERFORM Check-Write-Complete.
```

```
*
 DRIVER-X.
     EXIT.
*
 FINISH-UP.
     CALL "&CLOSEBMULTI" USING error-code, Sh..
     IF NOT No-Error
         GO TO finish-up.
     STOP RUN.
*
 GET-ADDRESS.
     DISPLAY "Enter Station Address: " UPON CONSOLE.
     ACCEPT device-address.
     CALL "&OPENBMULTI" USING error-code, device-address,
                              fss, pri, Sh, asBlk.
     IF NOT No-Error
         PERFORM print-error.
*
 GET-ADDRESS-X.
     EXIT.
*
 CHECK-READ-COMPLETE.
     IF NOT ReadDone
         GO TO CHECK-READ-COMPLETE.
*
 CHECK-WRITE-COMPLETE.
     IF NOT WriteDone
         GO TO CHECK-WRITE-COMPLETE.
*
 ERROR-CHECK.
     IF NOT No-Error
         PERFORM print-error.
*
 PRINT-ERROR.
     MOVE error-code TO ercdisplay.
     DISPLAY "BMULTI error ", ercdisplay UPON CONSOLE.
     STOP RUN.
*
 END-OF-JOB.
```

BASIC Echo Program (Using High-Level Interface)

This program echoes any text received back to the host system.

```
5     '************************************
7     '**                                **
10    '**     BASIC HLI Echo program     **
15    '**                                **
20    '************************************
110   DIM ASBlk% [7]
112   DIM Msg$ [4]
114   DIM Buf% [1024]
118   i% = 0
120   Erc% = 0
124   Daddr% = 0
125   fSys% = 0
126   Priority% = 127
127   TaskH% = 0
128   Optn% = 0
130   DevAdr$ = "aa"
132   sBuf% = 0
134   sBufMax% = 2048
135   ErcOk%  = 0
136   RcvStatus% = 0
137   XmtStatus% = 0
255   '
259   '************************************
260   '** Receive Status - BMulti         **
261   '************************************
262   Rec.Initial%     = 0
264   Receiving%       = 1
266   RecvFail%        = 2
268   Receive.Done%    = 16
270   Rec.XmnoEr%      = 17
272   Rec.DupXmno%     = 18
274   Rec.Trunc%       = 20
280   '
281   '************************************
282   '** Transmit Status - BMulti        **
283   '************************************
290   Xmit.Initial%    = 0
291   Transmiting%     = 1
292   Xmit.Failure%    = 2
293   Transmit.Done%   = 16
300   '
342   Msg$ [1] = " Bmulti Echo program"
350   Msg$ [2] = " Enter Address:  "
360   Msg$ [3] = " Begin Bmulti"
370   Msg$ [4] = " End Bmulti Echo"
380   messag$  = " Command Error"
390   '
```

```
392  '************************************
394  '**   Beginning of procedural code **
396  '************************************
398  '
400  PRINT Msg$ [1]
402  '
405  PRINT Msg$ [2]
410  INPUT DevAdr$
415  Daddr% = CVI(DevAdr$)
420  Erc% = OpenBMulti(Daddr%, fSys%, Priority%, PTR(TskH%),
     PTR(ASBlk%[1]))
422  IF Erc% <> ErcOk% THEN GOTO 1550
426  '
430  PRINT Msg$ [3]
435  '
440  Erc% = SetOptionBMulti(TskH%, Optn%)
442  IF Erc% <> ErcOk% THEN GOTO 1550
444  '
530  '******************
540  '** main loop     **
550  '******************
560  FOR i% = 1 TO 50
600  Erc% = ReadBMulti(TskH%, PTR(Buf%[1]), sBufMax%)
610  IF Erc% <> ErcOk% THEN GOTO 1550
620  RcvStatus% = ASBlk%[1] AND &H00FF
645  IF RcvStatus% <> Receive.Done% GOTO 620
650  '
1000 sBuf% = Buf%[1]
1010 Erc% = WriteBMulti(TskH%, PTR(Buf%[2]), sBuf%)
1020 IF Erc% <> ErcOk% THEN GOTO 1550
1022 XmtStatus% = ASBlk%[3] AND &H00FF
1024 IF XmtStatus% <> Transmit.Done% THEN GOTO 1022
1025 PRINT i%
1026 '
1120 NEXT i%
1122 '****************************
1124 '** end of main loop       **
1126 '****************************
1130 PRINT Msg$ [4]
1140 Erc% = ResetBMulti(TskH%)
1150 IF Erc% <> ErcOk% GOTO 1550
1151 Erc% = CloseBMulti(TskH%)
1152 IF Erc% <> ErcOk% GOTO 1550
1155 '
1159 STOP
1160 END
1170 '
1499 '************************************************
1500 '**   subroutine for displaying the erc         **
1501 '************************************************
1550 PRINT messag$, " ", Erc%
1600 END
```

FORTRAN Echo Program (Using High-Level Interface)

This program echoes any text received back to the host system.

```
$STORAGE: 2

      SUBROUTINE Error(int)
        CHARACTER*20 msg
        DATA msg/' Command Denied   '/
        WRITE (*,'(A,I6)') msg, int
        CALL Tmexit
      END

      PROGRAM echo
        IMPLICIT INTEGER*2 (a,b,c)
        EXTERNAL BMultO, BMultR, BMultW, BMultS, BMultT, BMultC
        INTEGER*2 erc, sBuf, sBufMx, ercOk, i, rcvSt, xmtSt,
     +            TskH, rcvInt, rcFail, rcvFin, erXmno,
     +            dupSeq, xmtInt, xmting, xmFail, xmtFin,
     +            fSys, trncEr, Prio, Optn, rcving

        CHARACTER*2 Buf
        CHARACTER*2 devadr
        CHARACTER*24 msg
        DIMENSION msg(5), Buf(1024)
        COMMON ASBlk(7)

        DATA  sBufMx/2045/, Prio/127/, Optn/0/
c
c  Error return codes
c
        DATA ercOk/0/
c
c  Receive Status - Bmulti
c
        DATA rcvInt/0/,
     +       rcving/1/,
     +       rcFail/2/,
     +       rcvFin/16/,
     +       erXmno/17/,
     +       dupSeq/18/,
     +       trncEr/20/
c
c  Transmit Status - BMulti
c
        DATA xmtInt/0/,
     +       xmting/1/,
     +       xmFail/2/,
     +       xmtFin/16/
c
```

```
      DATA msg(1)/' BMULTI echo program'/,
     +     msg(2)/' Enter Station Address: '/,
     +     msg(3)/' Begin BMULTI'/,
     +     msg(4)/' End BMULTI echo program'/.
c
c
c  Beginning of procedural code
c
c
       WRITE (*,'(A)') msg(1)

 1     WRITE (*,'(A)') msg(2)
       READ  (*,'(A2)') devadr
       erc = BMultO(devadr, fSys, Prio, TskH, ASBlk(1))
       IF (erc.NE.ercOk) CALL Error (erc)

       WRITE (*,'(A)') msg(3)

       CALL BMultS(TskH, Optn)
       IF (erc.NE.ercOk) CALL Error (erc)

       DO 10 i = 1, 50

          erc = BMultR(TskH, Buf(1), sBufMx)
          IF (erc.NE.ercOk) CALL Error (erc)
 2        rcvSt = ( MOD (ASBlk(1),256))
          IF (rcvSt.NE.rcvFin) GOTO 2

          sBuf = Buf(1)
          erc = BMultW(TskH, Buf(2), sBuf)
          IF (erc.NE.ercOk) CALL Error (erc)
 3        xmtSt = (MOD (ASBlk(3),256))
          IF (xmtSt.NE.xmtFin) GOTO 3
          WRITE (*,'(I6)') i

 10    CONTINUE

       WRITE (*,'(A)') msg(4)
 9990  erc = BMultT(TskH)
       IF (erc.NE.ercOk) GOTO 9990
 9999  erc = BMultC(TskH)
       IF (erc.NE.ercOk) GOTO 9999

       CALL Tmexit
     END
```

Pascal Terminal (Using High-Level Interface)

This program uses the high-level interface to implement a simple dumb terminal. Keyboard and screen code is shown along with data comm handling.

```
{$DEBUG-}
{$ENTRY-}

*PROGRAM  HLITerm;

TYPE
    String2   = STRING(2);
    pbType    = ADS OF BYTE;
    pwType    = ADS OF WORD;
    ppType    = ADS OF pbType;
    psType    = ADS OF String2;

CONST

 (* Miscellaneous*)
    banner          = 'B 20 Mini-Term (HLI)';
    sBanner         = 19;
    ErcOk           = 0;

VAR [PUBLIC]
    Erc             : WORD;
    Report          : WORD;
    th              : WORD;
    DevAdr          : String2;
    cMsg            : INTEGER;
    Msg             : ARRAY[0..79] OF BYTE;
    sBufMax         : INTEGER;
    sBuf            : INTEGER;
    Buff            : ARRAY[0..2047] OF CHAR;
    dummyPtr        : pbType;
    pVidSeg         : pbType;
    sMap            : WORD;
    nLines          : INTEGER;
    Key             : BYTE;
    current_col     : INTEGER;
    vid_col         : INTEGER;
    SdRet           : RECORD
        pSubParam   : psType;
        sSubParam   : WORD;
                      END;
    vHdw            : RECORD
        level       : BYTE;
        nLinesMax   : SINT;
        nColsNar    : BYTE;
        nColsWide   : BYTE;
                      END;
```

```
  ASBlk      : RECORD
    RcvStatus [00] : BYTE;

    RcvErc     [01] : WORD;
    fSelDen    [03] : BOOLEAN;
    XmtStatus  [04] : BYTE;
    XmtErc     [05] : WORD;
    Option     [07] : BYTE;
    fFastMsg   [08] : BOOLEAN;
    pFastMsg   [09] : pbType;
               END;

VALUE
    sBufMax         := 4096;
    sMap            := 16#0B6C;

!#####################################################
!#          System Common Procedures
!#####################################################
PROCEDURE Exit; EXTERN;

PROCEDURE ErrorExit (ercTerm : WORD); EXTERN;

FUNCTION  PosFrameCursor (iFrame : INTEGER;
                          iCol   : INTEGER;
                          iLine  : INTEGER) : WORD; EXTERN;

FUNCTION  PutFrameChars (iFrame : INTEGER;
                         iCol   : INTEGER;
                         iLine  : INTEGER;
                         pbText : pbType;
                         cbText : INTEGER) : WORD; EXTERN;

FUNCTION  ResetFrame ( iFrame : INTEGER) : WORD; EXTERN;

FUNCTION  ScrollFrame ( iFrame      : INTEGER;
                        iLineStart : INTEGER;
                        iLineMax   : INTEGER;
                        cLines     : INTEGER;
                        fUp        : BOOLEAN) : WORD; EXTERN;

!#####################################################
!#          Object Module Procedures
!#####################################################
FUNCTION  CloseBMULTI (Sh : BYTE) : WORD; EXTERN;

FUNCTION  OpenBMULTI (devAdr : String2;
                      fSys   : BOOLEAN;
                      Pri    : WORD;
                      pSh    : pbType;
                      pAsBlk : pbType) : WORD; EXTERN;

FUNCTION  ReadBMULTI (Sh   : BYTE;
                      pBuf : pbType;
                      sBuf : INTEGER) : WORD; EXTERN;

FUNCTION  ResetBMULTI (Sh  : BYTE) : WORD; EXTERN;
```

```
FUNCTION  WriteBMULTI (Sh    : BYTE;
                       pBuf  : pbType;
                       sBuf  : INTEGER) : WORD; EXTERN;

FUNCTION  RgParam (iParam    : WORD;
                   iSubParam : WORD;
                   pSdRet    : pbType) : WORD; EXTERN;

!#####################################################
!#          Procedural requests
!#####################################################
FUNCTION  InitCharMap (pMap : pbType;
                       sMap : WORD) : WORD; EXTERN;

FUNCTION  InitVidFrame (iFrame      : INTEGER;
                        iColStart   : INTEGER;
                        iLineStart  : INTEGER;
                        nCols       : INTEGER;
                        nLines      : INTEGER;
                        borderDesc  : BYTE;
                        bBorderChar : CHAR;
                        bBorderAttr : BYTE;
                        fDblHigh    : BOOLEAN;
                        fDblWide    : BOOLEAN) : WORD; EXTERN;

FUNCTION  QueryVidHdw (pBuf : pbType;
                       sBuf : WORD) : WORD; EXTERN;

FUNCTION  ReadKbdDirect (mode     : WORD;
                         pCharRet : pbType) : WORD; EXTERN;

FUNCTION  ResetVideo (nCols    : INTEGER;
                      nLines   : INTEGER;
                      fAttr    : BOOLEAN;
                      bSpace   : CHAR;
                      psMapRet : pbType) : WORD; EXTERN;

FUNCTION  SetScreenVidAttr (iAttr : WORD;
                            fOn   : BOOLEAN) : WORD; EXTERN;

!#####################################################
  PROCEDURE Check_Erc (Irk : WORD)  [PUBLIC];
!#####################################################
BEGIN
  IF (Irk <> 0) THEN ErrorExit (Irk);
END;

!#####################################################
  PROCEDURE Screen_setup  [PUBLIC];
!#####################################################
VAR
  BannerStart : INTEGER;.
BEGIN
  Check_Erc (QueryVidHdw (ADS vHdw, 4));

  nLines := vHdw.nLinesMax;
  Check_Erc (ResetVideo (80, nLines, FALSE,
                         ' ', ADS sMap));
```

```
!#####################################################
!##  Frame 0 is the line at the top of the screen.
!##  It has a solid thin border.
!#####################################################
  Check_Erc (InitVidFrame (0, 0, 0, 80, 1,
                           4, CHR(#0DA), 0, FALSE, FALSE));
!#####################################################
!##  Frame 1 is from line 3 to one line above the
!##  bottom of the screen.  It has the same border.
!#####################################################
  Check_Erc (InitVidFrame (1, 0, 2, 80, nLines - 3,
                           0, ' ', 0, FALSE, FALSE));
!#####################################################
!##  Frame 2 overlaps frame 1 completely, but also
!##  includes the last line on the screen.
!#####################################################
  Check_Erc (InitVidFrame (2, 0, 2, 80, nLines - 2,
                           0, ' ', 0, FALSE, FALSE));
  pVidSeg.s := 0;
  pVidSeg.r := 0;
  Check_Erc (InitCharMap (pVidSeg, sMap));
!##  Initiate video refresh  ##
  Check_Erc (SetScreenVidAttr (1, TRUE));
  Check_Erc (PosFrameCursor (2, 0, nLines - 3));
  current_col := 0;
  BannerStart := (80 - sBanner ) DIV 2;
  Check_Erc (PutFrameChars (0, BannerStart, 0,
                            ADS banner, sBanner));
END; (* PROCEDURE Screen_setup *)

!#####################################################
  PROCEDURE Process_Dcom_input  [PUBLIC];
!#####################################################
VAR
  i    : INTEGER;
BEGIN
  Check_Erc (ScrollFrame (1, 0, 255, 1, TRUE));
  Check_Erc (PosFrameCursor (1, 255, 255));
  Check_Erc (PosFrameCursor (2, 0, nLines - 3));
  vid_col := 0;
  IF sBuf > 0
    THEN
      FOR i := 0 TO sBuf - 1 DO
        BEGIN
          Check_Erc (PutFrameChars (1, vid_col, nLines - 4,
                                    ADS buff [i], 1));
          vid_col := vid_col + 1;
          IF (vid_col > 79).
            THEN
              BEGIN
                Check_Erc (ScrollFrame (1, 0, 255, 1, TRUE));

                Check_Erc (PosFrameCursor (1, 255, 255));
                vid_col := 0;
              END;
        END;
END; (* PROCEDURE Process_Dcom_input *)
```

```
!#####################################################
  PROCEDURE Process_Kbd_input  [PUBLIC];
!#####################################################
BEGIN
  IF (Key = 8)    !8=BACKSPACE
    THEN
      BEGIN
        IF (current_col = 79)
          THEN Check_Erc (PutFrameChars (2, 79,
                          nLines - 3, ADS ' ', 1));
        IF (current_col > 0)
          THEN current_col := current_col - 1;
        Check_Erc (PutFrameChars (2, current_col, nLines - 3,
                                  ADS ' ', 1));
        Check_Erc (PosFrameCursor (2, current_col, nLines - 3));
      END
    ELSE
      BEGIN
        Msg [current_col] := Key;
        Check_Erc (PutFrameChars (2, current_col, nLines - 3,
                                  ADS Key, 1));
        IF (current_col < 79)
          THEN current_col := current_col + 1;
        Check_Erc (PosFrameCursor (2, current_col, nLines - 3));
      END;
END; (* PROCEDURE Process_Kbd_input *)

!#####################################################
  PROCEDURE Active_state  [PUBLIC];
!#####################################################
!#   This is the main loop of the program.  It
!#   alternately checks the BMULTI report queue
!#   and the keyboard queue for activity.
!#####################################################
VAR
  loop1       : BOOLEAN;
  loop2       : BOOLEAN;

BEGIN
  WHILE TRUE DO
  BEGIN

    IF ASBlk.XmtStatus = 16
      THEN.
        BEGIN
          ASBlk.XmtStatus := 0;
          erc := ReadBMULTI (th, ADS sBuf, sBufMax);
        END;


    IF ASBlk.XmtStatus = 2
      THEN         BEGIN
          erc := ResetBMULTI (th);
          erc := WriteBMULTI (th, ADS Msg, cMsg);
        END;
```

```
    IF ASBlk.RcvStatus = 16
      THEN
        BEGIN
          ASBlk.RcvStatus := 0;
          Process_Dcom_Input;
          erc := ReadBMULTI (th, ADS sBuf, sBufMax);
        END;

    Erc := ReadKbdDirect (1, ADS Key);
    IF (Erc <> 602)
      THEN
          CASE Key OF

            4:  RETURN;  !FINISH key

            10, 27:       !RETURN, NEXT, and GO keys
               BEGIN
                 cMsg := current_col;
                 ERC := ResetBMULTI (th);
                 Erc := WriteBMULTI (th, ADS Msg, cMsg);
                 IF (Erc = ErcOk)
                   THEN
                     BEGIN
                       Check_Erc (ScrollFrame (2, 0, 255, 1, TRUE));
                       Check_Erc (PosFrameCursor (2, 0, nLines - 3));
                       current_col := 0;
                     END
               END

            OTHERWISE   Process_Kbd_Input;

          END;  (* CASE Key OF *)
  END;   (* WHILE TRUE DO *)

END;   (* PROCEDURE Active_state *)
!#######################################################
!#          MAIN PROGRAM
!#######################################################
BEGIN

!#######################################################
!#  Retrieve Device Address as either parameter 1 or 2,.
!#  depending on whether Run File command is used, or
!#  the program's own command.
!#######################################################

  Check_Erc (RgParam (1, 0, ADS SdRet));
  IF (SdRet.sSubParam <> 2)
    THEN  Check_Erc (RgParam (2, 0, ADS SdRet));
  DevAdr := SdRet.pSubParam?;

!#######################################################
!#  Initialize the video.
!#######################################################
  Screen_setup;
```

```
!#####################################################
!#  Log onto BMULTI with the Device Address.
!#####################################################
  Erc := OpenBMULTI (DevAdr, FALSE, 80, ADS th, ADS ASBlk);
  IF (Erc <> ErcOk)
    THEN ErrorExit (Erc);
  Erc := ReadBMULTI (th, ADS sBuf, sBufMax);

!#####################################################
!#  Enter an infinite loop.
!#####################################################
  Active_state;     !Does not return until FINISH is hit.

!#####################################################
!#  At termination, deallocate resources.
!#####################################################
  Erc := CloseBMULTI (th);
  Exit;
END.       (* PROGRAM HLITerm *)
```

Pascal Terminal (Using Low-Level Interface)

This program uses the low-level interface to implement a simple dumb terminal. Keyboard and screen code is shown along with data comm handling.

```
{$DEBUG-}
{$ENTRY-}

PROGRAM MiniTermBm;

TYPE
    String2   = STRING(2);
    pbType    = ADS OF BYTE;
    pwType    = ADS OF WORD;
    ppType    = ADS OF pbType;
    psType    = ADS OF String2;

CONST

 (* Command codes for new low-level interface *)
    Xfer_Rec_Bufc = 16#0001;
    Xfer_Xmt_Bufc = 16#0002;
    Offlinec      = 16#0003;
    Onlinec       = 16#0004;
    Idlec         = 16#0005;
    Fastsetc      = 16#0006;
    Receivec      = 16#0007;
    Transmitc     = 16#0008;
    Endsessionc   = 16#0009;
    Abortc        = 16#000A;
    Fastresetc    = 16#000B;
    Xmt_Big_Bufc  = 16#000C;

 (* Report codes *)
    No_report     = 16#0000;
    Receiving     = 16#0001;
    Rec_Grp_Sel   = 16#0002;
    Receive_Done  = 16#0003;
    Rdy_Xmt_Xfer  = 16#0004;
    Transmit_Done = 16#0005;
    Select_Denied = 16#0006;
    Receive_err   = 16#0007;
    Dup_seq_num   = 16#0008;
    Seq_num_er    = 16#0009;
    Transmit_err  = 16#000A;
    Internal_err  = 16#00FF;

 (* Erc return codes for new low-level interface *)
    ErcOk                = 16#0000;
    ErcInvalidCmd        = 16#8000;
    ErcTaskOverflow      = 16#8001;
    ErcCmdPending        = 16#8002;
    ErcReportPending     = 16#8003;
    ErcInvalidAddress    = 16#8004;
```

```
    ErcCmdDenied          = 16#8005;
    ErcBufferOverflow     = 16#8006;
    ErcInvalidReportRq    = 16#8007;
    ErcReadInProgress     = 16#8008;
    ErcWriteInProgress    = 16#8009;
    ErcBufferInUse        = 16#800A;
    ErcInvalidBufLength   = 16#800B;
    ErcOfflineDenied      = 16#800C;
    ErcOnlineDenied       = 16#800D;
    ErcIdleDenied         = 16#800E;
    ErcFastRdyDenied      = 16#800F;
    ErcXmtRdyDenied       = 16#8010;
    ErcRcvRdyDenied       = 16#8011;
    ErcXfrXmtDenied       = 16#8012;
    ErcXfrRcvDenied       = 16#8013;
    ErcEndSessDenied      = 16#8014;
    ErcNotOnline          = 16#8015;
    ErcStationOverflow    = 16#8016;
    ErcAddrIsGrpAddr      = 16#8017;
    ErcIncompleteMsg      = 16#8018;
    ErcInternalError      = 16#8019;
    ErcDupVirtualAdr      = 16#801A;
    ErcReconfiguration    = 16#801B;
    ErcEntryError         = 16#801C;
    ErcStationActive      = 16#801D;

 (* Miscellaneous*)
    banner           = 'B20 Mini-Term (LLI)';
    sBanner          = 19;

VAR [PUBLIC]
    Erc             : WORD;
    Report          : WORD;
    Sh              : byte;
    DevAdr          : String2;
    DevAdrRet       : String2;
    cMsg            : INTEGER;
    Msg             : ARRAY[0..79] OF BYTE;
    sBufMax         : INTEGER;
    sBuf            : INTEGER;
    Buff            : ARRAY[0..2047] OF CHAR;
    dummyPtr        : pbType;
    pVidSeg         : pbType;
    sMap            : WORD;
    nLines          : INTEGER;
    Key             : BYTE;
    current_col     : INTEGER;
    vid_col         : INTEGER;
    SdRet           : RECORD
        pSubParam : psType;
        sSubParam : WORD;
                    END;
    vHdw            : RECORD.
        level     : BYTE;
        nLinesMax : SINT;
        nColsNar  : BYTE;
        nColsWide : BYTE;
                    END;
```

```
VALUE
    sBufMax         := 4096;
    sMap            := 16#0B6C;

!#####################################################
!#          System Common Procedures
!#####################################################
PROCEDURE Exit; EXTERN;

PROCEDURE ErrorExit (ercTerm : WORD); EXTERN;

FUNCTION  PosFrameCursor (iFrame : INTEGER;
                          iCol   : INTEGER;
                          iLine  : INTEGER) : WORD; EXTERN;

FUNCTION  PutFrameChars (iFrame : INTEGER;
                         iCol   : INTEGER;
                         iLine  : INTEGER;
                         pbText : pbType;
                         cbText : INTEGER) : WORD; EXTERN;

FUNCTION  ResetFrame ( iFrame : INTEGER) : WORD; EXTERN;

FUNCTION  ScrollFrame ( iFrame      : INTEGER;
                        iLineStart : INTEGER;
                        iLineMax   : INTEGER;
                        cLines     : INTEGER;
                        fUp        : BOOLEAN) : WORD; EXTERN;

!#####################################################
!#          Object Module Procedures
!#####################################################
FUNCTION  BmOpenll (Addr      : string2;
                    pSh       : pbType;
                    fSys      : boolean;
                    pAddrRet  : pwType;
                    pbNode    : pbType;
                    cbNode    : word;
                    pbDesc    : pbType;
                    cbDesc    : word) : WORD; EXTERN;

FUNCTION BmReportTimeout (TaskH      : WORD;
                          pReportRet : pwType;
                          timeout    : WORD) : WORD; EXTERN;

FUNCTION BmReportWait (TaskH      : WORD;
                       pReportRet : pwType) : WORD; EXTERN;

FUNCTION BmReport (TaskH      : WORD;
                   pReportRet : pwType) : WORD; EXTERN;

FUNCTION BmCommand (TaskH   : WORD;
                    Comm    : WORD;
                    pbuffer : pbType;
                    sbuffer : INTEGER) : WORD; EXTERN;

FUNCTION  RgParam (iParam    : WORD;
                   iSubParam : WORD;
                   pSdRet    : pbType) : WORD; EXTERN;
```

```
!#####################################################
!#          Procedural requests
!#####################################################
FUNCTION  InitCharMap (pMap : pbType;
                       sMap : WORD) : WORD; EXTERN;

FUNCTION  InitVidFrame (iFrame      : INTEGER;
                        iColStart   : INTEGER;
                        iLineStart  : INTEGER;
                        nCols       : INTEGER;
                        nLines      : INTEGER;
                        borderDesc  : BYTE;
                        bBorderChar : CHAR;
                        bBorderAttr : BYTE;
                        fDblHigh    : BOOLEAN;
                        fDblWide    : BOOLEAN) : WORD; EXTERN;

FUNCTION  QueryVidHdw (pBuf : pbType;
                       sBuf : WORD) : WORD; EXTERN;

FUNCTION  ReadKbdDirect (mode     : WORD;
                         pCharRet : pbType) : WORD; EXTERN;

FUNCTION  ResetVideo (nCols    : INTEGER;
                      nLines   : INTEGER;
                      fAttr    : BOOLEAN;
                      bSpace   : CHAR;
                      psMapRet : pbType) : WORD; EXTERN;

FUNCTION  SetScreenVidAttr (iAttr : WORD;
                            fOn   : BOOLEAN) : WORD; EXTERN;

!#####################################################
  PROCEDURE Check_Erc (lrk : WORD)  [PUBLIC];
!#####################################################
BEGIN
  IF (lrk <> 0) THEN ErrorExit (lrk);
END;

!#####################################################
  PROCEDURE Screen_setup  [PUBLIC];
!#####################################################.
!#   Switches the video map back and forth.
!#####################################################
VAR
  BannerStart : INTEGER;
BEGIN
  Check_Erc (QueryVidHdw (ADS vHdw, 4));
  nLines := vHdw.nLinesMax;
  Check_Erc (ResetVideo (80, nLines, FALSE,
                         ' ', ADS sMap));
  Check_Erc (InitVidFrame (0, 0, 0, 80, 1,
                           4, '_', 0, FALSE, FALSE));
  Check_Erc (InitVidFrame (1, 0, 2, 80, nLines - 3,
                           0, ' ', 0, FALSE, FALSE));
  Check_Erc (InitVidFrame (2, 0, 2, 80, nLines - 2,
                           0, ' ', 0, FALSE, FALSE));
  pVidSeg.s := 0;
  pVidSeg.r := 0;
```

```
  Check_Erc (InitCharMap (pVidSeg, sMap));
  Check_Erc (SetScreenVidAttr (1, TRUE));
  Check_Erc (PosFrameCursor (2, 0, nLines - 3));
  current_col := 0;
  BannerStart := (80 - sBanner ) DIV 2;
  Check_Erc (PutFrameChars (0, BannerStart, 0,
                            ADS banner, sBanner));
END; (* PROCEDURE Screen_setup *)

!#####################################################
  PROCEDURE Process_Dcom_input  [PUBLIC];
!#####################################################
VAR
  i     : INTEGER;
BEGIN
  Check_Erc (ScrollFrame (1, 0, 255, 1, TRUE));
  Check_Erc (PosFrameCursor (1, 255, 255));
  vid_col := 0;
  FOR i := 0 TO sBuf - 1 DO
    BEGIN
      Check_Erc (PutFrameChars (1, vid_col, nLines - 4,
                                ADS buff [i], 1));
      vid_col := vid_col + 1;
      IF (vid_col > 79)
        THEN
          BEGIN
            Check_Erc (ScrollFrame (1, 0, 255, 1, TRUE));
            Check_Erc (PosFrameCursor (1, 255, 255));
            vid_col := 0;
          END;
    END;
END; (* PROCEDURE Process_Dcom_input *)

!#####################################################
  PROCEDURE Process_Kbd_input  [PUBLIC];
!#####################################################
BEGIN.
  IF (Key = 8)    !8=BACKSPACE
    THEN
      BEGIN
        IF (current_col = 79)
          THEN Check_Erc (PutFrameChars (2, 79,
                          nLines - 3, ADS ' ', 1));
        IF (current_col > 0)
          THEN current_col := current_col - 1;
        Check_Erc (PutFrameChars (2, current_col, nLines - 3,
                                  ADS ' ', 1));
        Check_Erc (PosFrameCursor (2, current_col, nLines - 3));
      END
    ELSE
      BEGIN
        Msg [current_col] := Key;
        Check_Erc (PutFrameChars (2, current_col, nLines - 3,
                                  ADS Key, 1));
        IF (current_col < 79)
          THEN current_col := current_col + 1;
        Check_Erc (PosFrameCursor (2, current_col, nLines - 3));
      END;
END; (* PROCEDURE Process_Kbd_input *)
```

```
!#####################################################
  PROCEDURE Active_state  [PUBLIC];
!#####################################################
!#   This is the main loop of the program.  It
!#   alternately checks the Bmulti report queue
!#   and the keyboard queue for activity.
!#####################################################
VAR
  loop1       : BOOLEAN;
  loop2       : BOOLEAN;

BEGIN
  WHILE TRUE DO
  BEGIN
    loop1 := TRUE;
    WHILE loop1 DO
    BEGIN
      Erc := BmReportTimeout (Sh, ADS Report, 2);
       IF (Erc = ErcOk)
         THEN
          CASE Report OF

            No_report:  loop1 := FALSE;

            Transmit_Done:
                Erc := BmCommand (Sh, Receivec, dummyPtr, 0);

            Rdy_Xmt_Xfer:
              Erc := BmCommand (Sh, Xfer_Xmt_Bufc,
                                ADS Msg, cMsg);

            Receive_Done, Dup_seq_num, Seq_num_err:
              BEGIN
                Erc := BmCommand (Sh, Xfer_Rec_Bufc,
                                  ADS sBuf, sBufMax);
                IF (Erc = ErcOk)
                  THEN
                    Process_Dcom_input;
                Erc := BmCommand (Sh, Receivec, dummyPtr, 0);
              END;
          END;     (* CASE Report OF *)
    END;    (* WHILE loop1 DO *)

    loop2 := TRUE;
    WHILE loop2 DO
    BEGIN
      Erc := ReadKbdDirect (1, ADS Key);
      IF (Erc = 602)
        THEN loop2 := FALSE
        ELSE
          BEGIN
            Erc := BmCommand (Sh, Idlec, dummyPtr, 0);
            CASE Key OF

            4:  RETURN;  !FINISH key

            10, 27:       !RETURN, NEXT, and GO keys
```

```
                BEGIN
                  cMsg := current_col;
                  Erc := BmCommand (Sh, Idlec, dummyPtr, 0);
                  Erc := BmCommand (Sh, Transmitc, dummyPtr, 0);
                  IF (Erc = ErcOk)
                    THEN
                      BEGIN
                        Check_Erc (ScrollFrame (2, 0, 255, 1, TRUE));
                        Check_Erc (PosFrameCursor (2, 0, nLines - 3));
                        current_col := 0;
                      Erc := BmCommand (Sh, Receivec, dummyPtr, 0);

                  END
                 END

              OTHERWISE   Process_Kbd_input;
             END;

           END;  (* CASE Key OF *)
    END;   (* loop2  *)
  END;   (* WHILE TRUE DO *)

END;   (* PROCEDURE Active_state *)
!#######################################################
!#          MAIN PROGRAM
!#######################################################
BEGIN

!#######################################################
!#  Retrieve Device Address as either parameter 1 or 2,
!#  depending on whether Run File command is used, or
!#  the program's own command.
!#######################################################
  Check_Erc (RgParam (1, 0, ADS SdRet));
  IF (SdRet.sSubParam <> 2)
    THEN  Check_Erc (RgParam (2, 0, ADS SdRet));
  DevAdr := SdRet.pSubParam?;

!#######################################################
!#  Initialize the video.
!#######################################################
  Screen_setup;

!#######################################################
!#  Log onto Bmulti with the Device Address.
!#######################################################
  Erc := BmOpenII (DevAdr, ADS Sh, FALSE, ads DevAdrRet,
                   ads nil, 0, ads 'LLITerm', 7);
  IF (Erc <> ErcOk)
    THEN ErrorExit (Erc);
  Erc := BmCommand (Sh, Onlinec, dummyPtr, 0);
```

```
!#####################################################
!#  Enter an infinite loop.
!#####################################################
  Active_state;     !Does not return until FINISH is hit.

!#####################################################
!#  At termination, deallocate resources.
!#####################################################
  REPEAT
    Erc := BmCommand (Sh, Idlec, dummyPtr, 0);
  UNTIL (Erc = ErcOk);
  Erc := BmCommand (Sh, Endsessionc, dummyPtr, 0);
  Exit;
END.        (* PROGRAM MiniTermBm *)
```

USASCII Code Charts

Figure B-1 **Universal Control Codes and Special Allocation of Codes to Implement with BMULTI protocol**

b7 b6 b5 → / BITS b4 b3 b2 b1 ↓ / COLUMN → / ROW ↓	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	0	1	2	3	4	5	6	7
0 0 0 0 / 0	NUL	DLE		0				POL
0 0 0 1 / 1	SOH	DC1		1				SEL
0 0 1 0 / 2	STX	DC2						
0 0 1 1 / 3	ETX	DC3						FSL
0 1 0 0 / 4	EOT	DC4						BSL
0 1 0 1 / 5	ENQ	NAK						
0 1 1 0 / 6	ACK	SYN						
0 1 1 1 / 7	BEL*	ETB						
1 0 0 0 / 8	BS	CAN						
1 0 0 1 / 9	HT	EM						
1 0 1 0 / 10	LF	SUB						
1 0 1 1 / 11	VT	ESC				[		
1 1 0 0 / 12	FF	FS		<				
1 1 0 1 / 13	CR	GS				]		
1 1 1 0 / 14	SO	RS						
1 1 1 1 / 15	SI	US						DEL

* CON (ALTERNATE CODE FOR CONTENTION)

Figure B-2 **USA Standard Code for Information Interchange (USASCII)**

b7 b6 b5 → / BITS b4 b3 b2 b1 ↓ / COLUMN → / ROW ↓	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	¦
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Hardware Requirements

To use BMULTI your BTOS workstation or XE 520 must be connected to a Burroughs mainframe. Typically the connection is through a direct line, a leased line, or the switched telephone network. Additional hardware is necessary to connect a BTOS workstation or XE 520 to any of these lines.

You may use either synchronous or asynchronous, two-wire or four-wire modems with BMULTI. Modems may be purchased from Burroughs or another vendor.

Because BTOS workstations and XE 520 communication hardware use the RS-232 interface, a modem or Burroughs TDI/Concatenation adaptor is required to connect them to a Burroughs Two-wire Direct Interface (TDI) network. The Burroughs DCA provides TDI according to the setting of the TDI/concatenation switch on the front panel.

If a BTOS workstation is to be connected to a modem line with other Burroughs terminals, a DCA is required to place the BTOS workstation anywhere other than last in the concatenation string. (This type of connection is not supported on the XE 520.) While set for concatenation, the adaptor turns on (applies a positive voltage to) pin 16 if downstream Request-to-Send is on. If the workstation turns off (applies a negative voltage to) pin 14, the adaptor blocks Request-to-Send from downstream terminals and Clear-to-Send from the modem. A second switch on the front of the adaptor sets the Rate Select and Select Standby signals to specified levels.

Note: *To install a BTOS workstation or XE 520 that is connected to an existing line, select a BTOS workstation or XE 520 modem that matches the host modem characteristics.*

In many systems, modem options are dictated by the conditions of the line to which the workstation or XE 520 is connected. Where other considerations permit, Burroughs recommends these modem option settings:

- Transmitter internally timed
- Four-wire operation
- Switched carrier
- No new synchronization

A double-male RS-232 extension cable must be used to connect the BTOS workstation or XE 520 to the modem. It should be a straight-through terminal-to-modem cable rather than the crossover (null modem) type.

RS-232C signals used in operation are shown in Table C-1. Those used in synchronous operation only are so marked.

Table C-1 **RS-232C Signals in Operation**

Pin number	Signal Name
1	Protective Ground
7	Signal Ground
2	Transmit Data
3	Receive Data
4	Request to Send (RTS)
5	Clear to Send (CTS)
6	Data Set Ready
8	Data Carrier Detect (Not used on BTOS workstations)
14	Block Downstream CTS (Not used on XE 520)
15	Transmit Clock (Sync only)
16	Sense Downstream RTS (Not used on XE 520)
17	Receive Clock (Sync only)
20	Data Terminal Ready

When using BTOS workstations in a concatenation environment (which requires the Burroughs DCA), upstream workstations must be turned on in order for downstream terminals or the BTOS workstations to communicate with the host.

Do not use the RS-232 cable shipped with the Burroughs DCA box for any connections other than from the DCA to the BTOS workstation or XE 520. Certain pins in the cable are used for other purposes by other Burroughs terminals. Using this cable with other terminals may cause unpredictable results.

Language Configuration

Before using FORTRAN, COBOL, or BASIC with BMULTI, you must regenerate the language interpreters and/or run time support modules. Use the Editor to add certain lines to the .asm file associated with your language (either COBOLGen.asm, BasGen.asm or ForGen.asm), assembling it, and either relinking the interpreter or, in the case of FORTRAN and compiled BASIC, linking the resultant object module with the object module that resulted from the compile.

The following lines must be added to the appropriate .asm file at the locations indicated by comments in each file, below the comment add new entries here:

BASIC (BasGen.asm)

```
%TableEntry(1,14,OPENBMULTI)
%TableEntry(1,8,READBMULTI)
%TableEntry(1,8,WRITEBMULTI)
%TableEntry(1,4,SETOPTIONBMULTI)
%TableEntry(1,2,RESETBMULTI)
%TableEntry(1,2,CLOSEBMULTI)
%TableEntry(1,24,BMOPENII)
%TableEntry(1,10,BMCOMMAND)
%TableEntry(1,6,BMREPORT)
%TableEntry(1,6,BMREPORTWAIT)
%TableEntry(1,8,BMREPORTTIMEOUT)
%TableEntry(1,24,BMGETSTATUS)
%TableEntry(1,6,BMQUERYHANDLE)
%TableEntry(1,6,BMSETXLATMODE)
%TableEntry(1,6,SELECTBMULTI)
%TableEntry(1,4,SETXLATMODEBMULTI)
```

COBOL (COBOLGen.asm)

Add these lines if you are using COBOL version 4.0 or earlier.

```
%TableEntry(0,w,OPENBMULTI,5,w,b,w,r,r)
%TableEntry(0,w,READBMULTI,3,b,r,w)
%TableEntry(0,w,WRITEBMULTI,3,b,r,w)
%TableEntry(0,w,SETOPTIONBMULTI,2,b,b)
%TableEntry(0,w,RESETBMULTI,1,b)
%TableEntry(0,w,CLOSEBMULTI,1,b)
%TableEntry(0,w,BMOPENII,8,w,r,b,x,r,w,r,w)
%TableEntry(0,w,BMCOMMAND,4,b,w,r,w)
%TableEntry(0,w,BMREPORT,2,b,x)
%TableEntry(0,w,BMREPORTWAIT,2,b,x)
%TableEntry(0,w,BMREPORTTIMEOUT,3,b,x,w)
%TableEntry(0,w,BMGETSTATUS,9,r,w,b,w,b,r,w,r,w)
%TableEntry(0,w,BMQUERYHANDLE,2,b,r)
%TableEntry(0,w,BMSETXLATMODE,3,b,b,b)
%TableEntry(0,w,SELECTBMULTI,2,r,w)
%TableEntry(0,w,SETXLATMODEBMULTI,3,b,b,b)
```

Add these lines if you are using COBOL version 5.0 or later.

```
%ExtrnRtn(OPENBMULTI,w,0,5,w,b,w,x,x)
%ExtrnRtn(READBMULTI,w,0,3,b,x,w)
%ExtrnRtn(WRITEBMULTI,w,0,3,b,x,w)
%ExtrnRtn(SETOPTIONBMULTI,w,0,2,b,b)
%ExtrnRtn(RESETBMULTI,w,0,1,b)
%ExtrnRtn(CLOSEBMULTI,w,0,1,b)
%ExtrnRtn(BMOPENII,w,0,8,w,x,b,y,x,w,x,w)
%ExtrnRtn(BMCOMMAND,w,0,4,b,w,x,w)
%ExtrnRtn(BMREPORT,w,0,2,b,y)
%ExtrnRtn(BMREPORTWAIT,w,0,2,b,w)
%ExtrnRtn(BMREPORTTIMEOUT,w,0,3,b,y,w)
%ExtrnRtn(BMGETSTATUS,w,0,9,r,w,b,w,b,r,w,r,w)
%ExtrnRtn(BMQUERYHANDLE,w,0,2,b,r)
%ExtrnRtn(BMSETXLATMODE,w,0,3,b,b,b)
%ExtrnRtn(SELECTBMULTI,0,w,2,r,w)
%ExtrnRtn(SETXLATMODEBMULTI,0,w,3,b,b,b)
```

FORTRAN (ForGen.asm)

Add these lines if you are using FORTRAN version 4.0 or earlier.

```
%TableEntry(OPENBMULTI,BMULTO,5,w,b,w,r,r)
%TableEntry(READBMULTI,BMULTR,3,b,r,w)
%TableEntry(WRITEBMULTI,BMULTW,3,b,r,w)
%TableEntry(SETOPTIONBMULTI,BMULTS,2,b,b)
%TableEntry(RESETBMULTI,BMULTT,1,b)
%TableEntry(CLOSEBMULTI,BMULTC,1,b)
%TableEntry(BMOPENII,BMOPII,8,w,r,b,r,r,w,r,w)
%TableEntry(BMCOMMAND,BMCMND,4,b,w,r,w)
%TableEntry(BMREPORT,BMREPT,2,b,r)
%TableEntry(BMREPORTWAIT,BMWAIT,2,b,r)
%TableEntry(BMREPORTTIMEOUT,BMTIME,3,b,r,w)
%TableEntry(BMGETSTATUS,BMGETS,9,r,w,b,w,b,r,w,r,w)
%TableEntry(BMQUERYHANDLE,BMQRYH,2,b,r)
%TableEntry(BMSETXLATMODE,BMSXLT,3,b,b,b)
%TableEntry(SELECTBMULTI,SELTBM,2,r,w)
%TableEntry(SETXLATMODEBMULTI,SXLTBM,3,b,b,b)
```

Add these lines if you are using FORTRAN version 5.0 or later.

```
%mediate(OPENBMULTI,BMULTO,5,w,b,w,r,r)
%mediate(READBMULTI,BMULTR,3,b,r,w)
%mediate(WRITEBMULTI,BMULTW,3,b,r,w)
%mediate(SETOPTIONBMULTI,BMULTS,2,b,b)
%mediate(RESETBMULTI,BMULTT,1,b)
%mediate(CLOSEBMULTI,BMULTC,1,b)
%mediate(BMOPEN,BMOPNI,8,w,r,b,r,r,w,r,w)
%mediate(BMOPENII,BMOPII,8,w,r,b,r,r,w,r,w)
%mediate(BMCOMMAND,BMCMND,4,b,w,r,w)
%mediate(BMREPORT,BMREPT,2,b,r)
%mediate(BMREPORTWAIT,BMWAIT,2,b,r)
%mediate(BMREPORTTIMEOUT,BMTIME,3,b,r,w)
%mediate(BMGETSTATUS,BMGETS,9,r,w,b,w,b,r,w,r,w)
%mediate(BMQUERYHANDLE,BMQRYH,2,b,r)
%mediate(BMSETXLATMODE,BMSXLT,3,b,b,b)
%mediate(SELECTBMULTI,SELTBM,2,r,w)
%mediate(SETXLATMODEBMULTI,SXLTBM,3,b,b,b)
```

BTOS Request Codes for BMULTI

BMULTI uses request codes -1 through -9, and five loadable request codes. All system software releases numbered 4.0 and higher support requests -1 through -7. In addition, BTOS 5.0 supports request -8, and BTOS 6.0 supports requests -8 and -9. System releases numbered 2.3 through 4.0 support request codes -1 and -2 only. System software releases 1.3 and lower do not support BMULTI. To upgrade a lower system software release to full functionality for BMULTI, use BTOS Customizer.

Add the following entries near the end of the file "Request.asm", in the user request code table:

```
%UsrRequest(-1,DCCommand,exchNotInstalled,0000h,3,1,1,
                    %( %illegal )
                    %( %norouting ))

%UsrRequest(-2,DCReport,exchNotInstalled,0000h,2,0,1,
                    %( %illegal )
                    %( %norouting ))

%UsrRequest(-3,BMCommand,exchNotInstalled,000h,4,1,1,
                    %( %illegal )
                    %( %norouting ))

%UsrRequest(-4,BMReport,exchNotInstalled,000h,2,0,1,
                    %( %illegal )
                    %( %norouting ))

%UsrRequest(-5,BMQuery,exchNotInstalled,000h,0,0,1,
                    %( %illegal )
                    %( %norouting ))

%UsrRequest(-6,BMClear,exchNotInstalled,000h,2,0,0,
                    %( %none )
                    %( %norouting ))

%UsrRequest(-7,BMPurge,exchNotInstalled,000h,2,0,0,
                    %( %none )
                    %( %norouting ))

%UsrRequest(-8,BMChangeUser,exchNotInstalled,000h,2,0,0,
                    %( %none )
                    %( %norouting ))

%UsrRequest(-9,QuietBMULTI,ExchNotInstalled,000h,2,0,0,
                    %( %none )
                    %( %norouting ))

%UsrRequest(-10,DCReserve,0,000h,0,0,0,
                    %( %illegal )
                    %( %norouting ))
```

The tables for termination requests, workstation abort requests, change user number requests, and swapping requests follow the user request code table. Add these entries:

```
%TerminationRequest(-6)        ; Request code for BMClear

%WsAbortRequest(-7)            ; Request code for BMPurge

%ChgUserNumRequest(-8)         ; ChangeUserNumBMULTI

%SwappingRequest(-9)           ; QuietBMULTI
```

Note: Unisys reserves user request codes -10 through -32 for future development. In particular, user request code –10 is reserved for future BMULTI development.

The file Request.asm must then be assembled as described in the *System Programmer's Guide, Volume 1*, for creating a new operating system.

The current version of BMULTI also uses the loadable request codes 0D02Ah, 0D02Bh, 0D02Dh, and 0D02Eh. These appear in the file "Request.F.Sys". The definitions are:

```
%Request(0D02Ah,(BmReport),ExchNotInstalled,0000h,2,0,1,
        %( %illegal )
        %( %rFh ))

%Request(0D02Bh,(BmOpenII),ExchNotInstalled,0000h,6,2,2,
        %( %illegal )
        %( %DevSpec %OpenFh ))

%Request(0D02Dh,(BmCommand),ExchNotInstalled,0000h,6,1,1,
        %( %illegal )
        %( %rFh ))

%Request(0D02Eh,(BmClose),ExchNotInstalled,0000h,2,0,0,
        %( %illegal )
        %( %rFh %CloseFh ))
```

Troubleshooting Notes

Rebooting the Master

As is the case in all BTOS environments, when you reboot a BTOS master that is running an application, such as BMULTI, you must also reboot all cluster workstations.

Receive Buffer Overflow

You are responsible for creating a receive buffer of adequate size. If the buffer does overflow (i.e., RcvStatus field in ASM equals 14), you should issue a delay of approximately 1 second or issue a ResetBMULTI prior to issuing another ReadBMULTI.

B-NET Support

The LLI and HLI support request routing over B-NET. A properly written application linked with the LLI or HLI can access a copy of BMULTI that is running on a remote B-NET node.

If you are running across B-NET, the B-NET install parameter values may need to be changed. In the Net Agent install form, there is a parameter:

```
[Max # client requests queued (default = 8)]
```

In the Net Server install form, there is a parameter:

```
[Max # data buffers (default = 4)]
```

The values used for these should be at least:

```
2 + (3 * (# BMULTI addresses in use))
```

in each case. This is similar to the XBLK/YBLK calculations.

This applies only to BMULTI addresses being used across the Net.

Earlier BMULTI Versions

The current system services support all older libraries as well as the current library with one exception: the procedures BmQuery and BmIdentify have been deimplemented. Any application using these procedures should be rewritten using the new procedure BmGetStatus and then recompiled and relinked using the current BMULTI library. In addition, any application linked with the current library will also require the current system service. Finally, the procedure BmOpen should be replaced with the procedure BmOpenII in any application being relinked with the current version of BMULTI. Otherwise, the application cannot take advantage of the newest performance enhancements.

BmOpenII, VADs, and Existing Applications

Applications that call BmOpenII with the new "any open VAD" option must be introduced carefully into existing installations, especially if they are being mixed with older applications with fixed addresses. The new application may request and be granted an unused VAD that is customarily used by another application, with the result that the operator cannot initiate the other application.

VADs and Sophisticated NDLs

If VADs are used on systems with sophisticated Network Definition Languages, they may cause a noticeable performance degradation on the data comm line. See Appendix H for more details.

Choosing the Channel to Use with the Data Comm Expander Module

You must install the Data Comm Expander (DCX) server before you install BMULTI if you have configured BMULTI to use channels 1A, 1B, 1C, 1D, 2A, 2B, 2C, or 2D. If this is not done, BMULTI will fail to install (with an erc of 60).

Response Time

BMULTI allows up to 64 addresses to be used by applications at one time. However, response time varies according to the number of addresses in use, as well as with the amount of data traffic generated by those applications. For example, you may be satisfied with the response time if you are running ten copies of a low-traffic, 3-address application. However, even ten copies of the MT 983 emulator may cause an unsatisfactory response time for your requirements and your application.

Configurable Delays

The configurable delays offered by BMULTI are upwardly variable because certain processes in the operating system have a higher priority than BMULTI. The actual delay is never less than that with which BMULTI is configured; however, it may be more.

Transmission Blocks

Applications can require a maximum of three transmission blocks per address. The number of requests an application will usually have outstanding is unchanged, however there is a new maximum of three. Master operating systems may not operate optimally if the number of XBlks (and YBlks and ZBLks for the XE 520) is not at least three times the number of BMULTI addresses to be used. (On the XE 520, these are divided between the cluster processors appropriately.)

Too few XBlks results in poor performance. Occasionally, however, a deadlock situation can arise. In this situation, all existing XBlks hold requests that cannot be responded to unless more requests are made. However, more requests cannot be made because all XBlks are in use.

Another problem caused by having too few XBlks on the Master occurs if cluster stations cannot issue Abort requests to return to Idle from Transmitting (for example, when using BTE, an operator presses the LOCAL key while the F10 LED is lit).

Configuration Parameters

If BMULTI does not operate, make sure that the values for baud rate or time delay are correct for the host you are operating as well as for BMULTI. Although you may have entered values that are accepted by BMULTI, they may not be correct for your current host.

Status Code 32786, "Transfer Xmt Buffer Command Denied"

Status code 32786 occasionally occurs when the Transfer Transmit Buffer command is issued. This usually happens because the application has not checked the report queue often enough, causing it to become too deep. To avoid this problem, have the application empty the queue before issuing the SetTransmitReady command.

Put as little code as possible between checking for a report of ready for transmit buffer and issuing the command transfer transmit buffer, and also put as little code as possible amid BTOS calls.

To recover from this problem, issue an Idle command and reinitiate the transmit sequence.

Status Codes Generated by Low-Level Interface

Error Code (Hex)	Error Code (Decimal)	Explanation
3C	60	BMULTI cannot be installed on a four-port expander because the four-port server is not installed.
8000	32768	Invalid command was issued to BMULTI.
8001	32769	Task overflow. The multitasking interface cannot handle more than three device addresses.
8002	32770	Command pending. Only one BMULTI command can be outstanding at any time per address.
8003	32771	Report pending. Only one BMULTI report request can be outstanding at any time for an address.
8004	32772	Invalid BMULTI address.
8005	32773	Command denied (Mp Interface).
8006	32774	Buffer overflow.
8007	32775	Invalid report request.
8008	32776	Read in progress.
8009	32777	Write in progress.
800A	32778	A request for a large buffer transfer cannot be honored at this time for want of buffer space. Try again later.
800B	32779	Invalid buffer length for this command.
800C	32780	Offline command denied.
800D	32781	Online command denied.
800E	32782	Idle command denied.

Error Code (Hex)	Error Code (Decimal)	Explanation
800F	32783	Fast Ready command denied.
8010	32784	Transmit Ready command denied.
8011	32785	Receive Ready command denied.
8012	32786	Transfer Xmt Buffer command denied.
8013	32787	Transfer Rcv Buffer command denied.
8014	32788	End session command denied.
8015	32789	Station not online.
8016	32790	Station overflow.
8017	32791	Device address clashes with group address.
8018	32792	The buffer received by the application is only part of the full message.
8019	32793	Internal error (Report to the system administrator).
801A	32794	Duplicate virtual address.
801B	32795	BMULTI locked for reconfiguration.
801C	32796	Entry error.
801D	32797	Station active. Context Manager cannot swap tasks at this time.
801E	32798	BMULTI was not deinstalled because the operating system is single-partition.
801F	32799	BMULTI was not deinstalled, either because the Deinstall utility was not executed in the primary application partition (Context Manager installed) or because the Deinstall utility was not executed on the machine on which BMULTI is installed.
8020	32800	Line monitor facility is in use.

Error Code (Hex)	Error Code (Decimal)	Explanation
8021	32801	Data lost while monitoring.
8022	32802	BMULTI was not deinstalled because stations are active.
8023	32803	An attempt to call BmOpenII, requesting an unused virtual address, failed because there are no more unused virtual addresses.
8024	32804	Inconsistency between BMULTI and BmZip, BmConfig, or BmStatus. If you have configuration files created with release level 4.0 or 5.0 of BMULTI, you must open them, make any new parameter selections (e.g., TDI?) and close them using the BMULTI 6.0 configurator (see "Configuring BMULTI" in Section 3).
8025	32805	Change of extended character translation mode attempted by BmSetXlatMode or SetXlatModeBMULTI when station not idle, or station online but not fast ready.
8026	32806	Invalid iRcvMode or iXmitMode specified for BmSetXlatMode or SetXlatModeBMULTI.
8027	32807	Character translation file incomplete or in unexpected format.
8028	32808	Data contents of character translation file not valid ASCII for 0-9, A-F, a-f, or space (30h-39h, 41h-46h, 61h-66h, or 20h).
8029	32809	BmGetStatus (iRecordLen * nStation) exceeds specified buffer size.
802A	32810	The number of stations or IO buffer size cannot be reconfigured dynamically.
802B	32811	BMULTI has already been installed.
802C	32812	An invalid node specification was passed to SelectBMULTI.
802D	32813	The report queue of the station requesting the report has overflowed and the contents of the queue are now meaningless.

Host/NDL Requirements

Mainframe/NDL Timeout Values

BMULTI is inherently slower than a Burroughs terminal because Burroughs terminals have processors dedicated to data comm processing. Therefore, host timeout and transmit delay parameters must be set to higher values when running networks that include BMULTI stations. The recommended minimal host timeout value for B 25 or XE 520 cluster configurations is 1000 milliseconds (1 second). A 25 to 50 millisecond range is recommended for the transmit delay. These settings may require modification by the user depending upon the actual performance of the network. In networks where all applications are linked with BMULTI 6.0 or higher, host timeout values can be set as low as 50 milliseconds. Applications must use the high-level or low-level interface to take advantage of the performance improvements of the 6.0 and higher libraries.

Virtual Addresses and Network Definition Languages

Virtual addresses (VADs), which are optional, should be used only with unsophisticated Network Definition Languages (NDLs). Acceptable NDLs are typically ones used on CMS machines and Burroughs small systems. The NDLs used in these environments send poll sequences to inactive stations until a timeout or the number of retries is exceeded. After either of these conditions becomes true, that station is removed from the poll sequence and can be reactivated only through host operator's intervention. This is the type of environment in which VADs are particularly useful.

Indiscriminate use of this feature on systems with sophisticated NDLs can cause a noticeable performance degradation on the data comm line. This is more noticeable when a large number of virtual addresses are defined in the BMULTI configuration file. The NDLs used on Burroughs large systems (e.g., B7900) and the new A Series mainframes preclude the use of virtual addresses. NDLs on these systems poll inactive addresses less frequently than active addresses, so that the line is not overloaded with unnecessary polls and selects. Do not use virtual addresses in these environments; if a station becomes active, the host eventually starts polling that terminal without any additional operator intervention.

BMULTI Configuration Worksheet

To the Site Administrator or Host Systems Administrator: You might find this tear-out worksheet a convenient way to supply operators with BMULTI configuration parameters. Anyone familar with BTOS workstations can use the Configurator (described in Section 3) to enter the values from this sheet. If you need to report a problem, attach this page to a Field Communication Form (FCF) and send it to the appropriate Resource Control Center or Customer Support Center.

Screen Message	Preset Values	Possible Values	Change to:	Notes
Group Poll Address	dv	020h—07Fh	_____	
Group Select Character	r	020h—07Fh	_____	
Sync or Async	A	'A' or 'S'	_____	
Channel	0B	A, B, C, D 0A,0B,0C,0D 1A,1B,1C,1D 2A,2B,2C,2D	_____ _____ _____ _____	
Baud Rate	09600	110—19200	_____	
Transmission Numbering	0	0,1,2,3,4,5	_____	
RTS-CTS delay	0	0—255	_____	
XMT-RCV delay	0	0—255	_____	
RTS HOLD delay	0	0—255	_____	
Anything Downstream?	N	Y or N	_____	
TDI on IDS?	N	Y or N	_____	
Number of stations	16	1-64	_____	
Buffer size	2048	16-4096	_____	
Translation Table Name	[Sys]<Sys>BmultiXlat.Sys		_____	

ring buffer, 5-4
RTS, 7-10 to 7-17
RTS-CTS delay, 3-1, 3-3
RVI (reverse interrupt) character, 7-11, 7-13, 8-1 to 8-18

S

SEL (select) character, 8-1 to 8-18
select (*see also* broadcast, fast, and group select)
  by host computer, 7-3, 7-5, 7-7, 7-8, 7-9
  denied, 7-5, 7-7, 7-8, 7-9
sequence number, 7-4
set fast ready command, 7-3, 7-5, 9-5, 10-2
SetOptionBMULTI, 9-5
SOH (Start of Header) character, 8-1 to 8-18
station:
  address, 7-3,
  control,
  downstream,
  handle,
  record format, 10-8
  reserving station address, 7-3
state change, 7-1 to 7-17
status codes generated by low-level interface, G-1
status monitor, 4-1
STX (start of text) character, 8-1 to 8-18
synchronous data communication, 8-1
SYN (Synchronous Idle) character, 8-1 to 8-18

T

TDI/concantenation adapter (*See* DCA Box)
timeout:
  BmReportTimeout, 10-1
  host, 1-1, 10-1
  NDL (Network Definition Language), H-1
  no response, 8-3
  VADs, 6-3
transfer receive buffer command, 7-4
transfer transmit buffer command, 7-4, 7-10, 10-1, F-4
transmission:
  asynchronous, 1-1, 8-1
  blocks, F-3
  general, 1-1, 7-4, 7-?, 8-1 to 8-18, 9-6
  synchronous, 1-1, 8-1
transmission numbering:
  alternating, 8-2
  sequential, 8-2

transmit:
buffer, 7-4, 7-8, 7-9, 10-1, 10-2, 10-4, F-4
command, 7-1 to 7-17, 9-6
done, 7-11, 7-12, 10-4
error, 7-4, 7-11, 7-13, 10-5
ready command, 7-13
troubleshooting, F-1

V

Virtual addresses (VADs), 6-3, F-2

W

WriteBmulti, 9-6

X

XMno (*see also* transmission number), **3-3, 8-1 to 8-18**

Examples of Shift-In and Shift-Out Action

The following examples clarify the actions occurring in the transmit and receive modes selected by the BmSetXlatMode and SetXlatModeBMULTI procedures.

BTOS Receiving Format

Original Data Stream	Data As Received by BTOS		Resultant User Buffer		
Position	**ASCII**	**Hex**	**Mode 0**	**Mode 1**	**Mode 2**
1	A	41	41	41	41
2	ESC	1B	1B		
3	SO	0E	0E		
4	B	42	42	C2	C2
5	ESC	1B	1B	9B	9B
6	3	33	33	B3	B3
7	1	31	31	B1	B1
8	ESC	1B	1B	9B	9B
9	'	27	27	A7	A7
10	.	2E	2E	AE	AE
11	ESC	1B	1B		
12	SI	0F	0F		
13	C	43	43	43	43
14	D	44	44	44	44
15	SO	0E	0E	0E	
16	E	45	45	45	C5
17	SI	0F	0F	0F	
18	3	33	33	33	33

In Mode 1, ESC SO, and ESC SI are not buffered. In Mode 2, ESC, ESC SO, ESC SI, SO, and SI are not buffered.

BTOS Transmitting Format

Original Data Stream	Data As Received by BTOS		Resultant User Buffer		
Position	**ASCII**	**Hex**	**Mode 0**	**Mode 1**	**Mode 2**
1	a	61	61	61	61
2	space	20	20	20	20
				1B	ESC
				0E	SO
3	n	EA	6A	6A	6A
				1B	ESC
				0F	SI
4	f	66	66	66	66
5	l	6C	6C	6C	6C
6	u	7D	7D	7D	7D
7		7E	7E	7E	7E
8	e	65	65	65	65
9	1	31	31	31	31
10	12	32	32	32	32
				1B	
				0E	0E
11	a	D3	53	53	53
12	e	DF	5F	5F	5F

Translation Tables

The character sets for various languages supported by the BTOS Systems are not the same as those on the Burroughs mainframes. This is true for standard characters (00h - 7Fh) as well as for extended characters (in the 80h - FEh range). As a consequence, some translation between character sets is necessary, both before transmission and when received. To accomplish this, BMULTI uses a translation table that is loaded from a disk file (specified in the BMULTI configuration file). The default translation file is called BmultiXlat.sys, and is the USA version.

Bmulti performs the translation of the corporate standard character set to the local BTOS character set, and vice versa. Characters that appear in only one of the character sets are translated to be a suitable character when possible, otherwise a question mark is substituted. Characters in conflict with the local BTOS character set are remapped above 80h. The template in this appendix shows the mapping for the seven different languages supported by the BTOS terminal emulators.

There are really two character translation tables built into the specified translation file. The first table is used to remap 256 characters from the mainframe to the BTOS, and the other is used to remap 256 characters from the BTOS to the host. The user should never remap the ETX character (03h) to any other value, or remap any other character to be an ETX. In general, it is not advisable to remap any poll/select protocol characters. BMULTI does not apply the translation tables to its own protocol.

This release of BMULTI can use only a single translation file at any given time, and it is shared by all users of that copy of BMULTI. BMULTI loads this translation table file (specified in the configuration file) from disk when it is installed, but the table can be changed dynamically by specifying a new translation table file using the Reconfigure BMULTI command. The user should be aware that this will change the translation table for all applications. Any translation table file to be used by BMULTI must reside on disk.

The format of a translation table file on disk is that of an editable text file. These table files are a matrix of hex character values separated by spaces, as shown in the supplied template. Comments can also be entered in this file. Comments and data alternate between pairs of colon delimiters. The file is always assumed to begin with a comment, and everything prior to the first colon is bypassed. When BMULTI is installed or reconfigured, the tables are loaded into memory. In the process, comments and delimiters are removed, and the two byte ASCII data entries are packed into a single byte format (one hexadecimal byte per character). This results in a 256 byte transmit translation table and a 256 byte receive translation table. The 3Fh character in the template is a question mark, used when no reasonable cross-mapping of characters exists.

If there is an odd number of data bytes (anything other than spaces) between colons, or if the end of file is encountered before the construction of the tables is complete, an error InvalidFileFormat is returned. If data field characters outside the ranges of 30h–39h (0–9) inlusive or 41h–46h (A–F) inclusive are encountered, an error InvalidFileData is returned.

Translation Template BTOS USA Standard Character Set to T 27 Version 1 (USA) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	3F	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	3F	3F	3F	3F	:
3:	03	13	23	33	43	53	63	73	3F	3F	3F	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	3F	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	3F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	3F	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	3F	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	3F	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	3F	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	3F	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	41	42	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	61	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	41	3F	3F	3F	3F	3F	:
3:	03	13	23	33	43	53	63	73	3F	3F	61	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	4F	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	6F	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	4F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	6F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	55	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	75	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	63	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	65	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	65	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	65	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

Translation Template BTOS Canadian Standard Character Set to T 27 Version 1/A (Canada) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	BB	4F	6E	AA	6F	:
1:	01	11	21	31	41	51	61	71	3F	3F	41	47	4F	A0	3F	6F	:
2:	02	12	22	32	42	52	62	72	3F	3F	41	49	53	61	3F	73	:
3:	03	13	23	33	43	53	63	73	3F	3F	B4	49	53	A1	3F	73	:
4:	04	14	24	34	44	54	64	74	3F	3F	41	B0	3F	61	A6	42	:
5:	05	15	25	35	45	55	65	75	3F	3F	41	49	50	61	3F	70	:
6:	06	16	26	36	46	56	66	76	3F	3F	41	BC	AF	61	AB	A8	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	55	3F	3F	75	:
8:	08	18	28	38	48	58	68	78	3F	3F	43	4A	B9	63	3F	A9	:
9:	09	19	29	39	49	59	69	79	3F	3F	B2	4C	BE	A2	3F	AC	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	43	4E	57	63	3F	77	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	44	3F	59	3F	3F	79	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	4F	59	3F	3F	79	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	AE	4F	59	A4	3F	79	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	BA	B1	5A	A3	A7	7A	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	B5	4F	3F	A5	3F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	D1	B4	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	D3	BE	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	D9	A9	3F	3F	3F	3F	:
3:	03	13	23	33	43	53	63	73	3F	3F	DE	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	DD	A3	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	DF	AF	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	E4	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	EE	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	F6	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	F8	C8	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	E0	AE	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	E6	B0	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	F9	B6	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	A1	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	AD	C9	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	C6	3F	3F	3F	3F	3F	:

Translation Template BTOS United Kingdom Standard Character Set to T 27 Version 2 (UK) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	3F	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	3F	3F	3F	3F	:
3:	03	13	B0	33	43	53	63	73	3F	3F	3F	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	3F	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	3F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	3F	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	3F	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	3F	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	3F	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	3F	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	23	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	7C	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	60	3F	3F	3F	3F	:
3:	03	13	3F	33	43	53	63	73	3F	3F	61	27	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	61	5E	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	61	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	63	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	65	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	65	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	65	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	69	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	6F	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	6F	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	75	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	75	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	75	3F	3F	3F	3F	3F	:

Translation Template BTOS Netherlands Standard Character Set to T 27 Version 2 (UK/Belgium/Italy) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	3F	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	3F	3F	3F	3F	:
3:	03	13	A4	33	43	53	63	73	3F	3F	3F	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	3F	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	3F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	3F	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	3F	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	3F	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	3F	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	3F	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	3F	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	5E	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	3F	3F	3F	3F	:
3:	03	13	3F	33	43	53	63	73	3F	3F	66	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	23	60	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	27	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	3F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	61	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	65	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	65	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	6F	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	75	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	61	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	65	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	75	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	65	3F	3F	3F	3F	3F	:

Translation Template BTOS South African Standard Character Set to T 27 Version 2 (UK/Belgium/Italy) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	3F	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	3F	3F	3F	3F	:
3:	03	13	B0	33	43	53	63	73	3F	3F	3F	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	3F	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	3F	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	3F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	3F	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	3F	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	3F	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	3F	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	3F	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	5E	23	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	27	3F	3F	3F	3F	3F	:
3:	03	13	3F	33	43	53	63	73	3F	3F	60	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	65	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	65	6E	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	6F	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	6F	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	61	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	65	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	65	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	75	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	61	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	75	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

Translation Template BTOS German Standard Character Set to T 27 Version 5/A (Germany) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	B5	50	60	70	3F	3F	3F	45	A1	6E	65	A4	:
1:	01	11	21	31	41	51	61	71	3F	3F	41	47	4F	A7	67	6F	:
2:	02	12	22	32	42	52	62	72	3F	3F	41	49	53	61	69	73	:
3:	03	13	23	33	43	53	63	73	3F	3F	41	49	53	AB	69	73	:
4:	04	14	24	34	44	54	64	74	3F	3F	41	49	3F	61	AD	AB	:
5:	05	15	25	35	45	55	65	75	3F	3F	A0	49	50	A3	69	70	:
6:	06	16	26	36	46	56	66	76	3F	3F	41	49	55	61	69	A9	:
7:	07	17	27	37	47	57	67	77	3F	3F	3F	3F	55	3F	3F	75	:
8:	08	18	28	38	48	58	68	78	3F	3F	43	4A	55	63	6A	AF	:
9:	09	19	29	39	49	59	69	79	3F	3F	43	4C	A2	63	6C	A5	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	43	4E	57	63	63	77	:
B:	0B	1B	2B	3B	4B	A0	6B	A3	3F	3F	44	3F	59	3F	3F	79	:
C:	0C	1C	2C	3C	4C	A1	6C	A4	3F	3F	3F	4F	59	64	6F	79	:
D:	0D	1D	2D	3D	4D	A2	6D	A5	3F	3F	45	4F	59	A8	6F	79	:
E:	0E	1E	2E	3E	4E	5E	6E	A6	3F	3F	45	4F	5A	AA	AE	7A	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	45	4F	3F	AC	6F	3F	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	3F	50	60	70	3F	3F	5B	60	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	5C	27	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	5D	5E	3F	3F	3F	3F	:
3:	03	13	23	33	43	53	63	73	3F	3F	7B	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	7C	3F	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	7D	40	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	7E	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	D1	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	DD	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	F6	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	DE	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	3F	6B	3F	3F	3F	D3	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	3F	6C	3F	3F	3F	DF	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	3F	6D	3F	3F	3F	E4	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	3F	3F	3F	EE	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	F8	3F	3F	3F	3F	3F	:

Translation Template BTOS French Standard Character Set to T 27 Version 43 (French Word Processor) Character Set

BTOS Receive Translation Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	3F	B1	41	44	A0	61	:
1:	01	11	21	31	41	51	61	71	3F	3F	3F	3F	41	4E	61	6E	:
2:	02	12	22	32	42	52	62	72	3F	3F	3F	3F	41	4F	A1	6F	:
3:	03	13	23	33	43	53	63	73	3F	3F	B2	3F	41	4F	61	6F	:
4:	04	14	24	34	44	54	64	74	3F	3F	3F	3F	41	4F	A2	AA	:
5:	05	15	25	35	45	55	65	75	3F	3F	3F	3F	41	4F	61	6F	:
6:	06	16	26	36	46	56	66	76	3F	3F	7C	3F	41	4F	3F	AB	:
7:	07	17	27	37	47	57	67	77	3F	3F	B0	3F	3F	3F	A3	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	B4	3F	43	45	A5	6F	:
9:	09	19	29	39	49	59	69	79	3F	3F	3F	3F	45	55	A4	AC	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	3F	3F	45	55	A6	75	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	3F	3F	45	55	A7	AD	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	3F	3F	45	55	69	AE	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	3F	3F	49	59	69	79	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	3F	3F	49	70	A8	70	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	2D	3F	49	42	A9	79	:

BTOS Transmit Translation Table

Extended character translations are made to closest related character (e.g., an "a umlaut" is translated to a plain "a"). A question mark (3Fh) is used when there is no suitable character.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0:	00	10	20	30	40	50	60	70	3F	3F	E0	A7	3F	3F	3F	3F	:
1:	01	11	21	31	41	51	61	71	3F	3F	E2	B0	3F	3F	3F	3F	:
2:	02	12	22	32	42	52	62	72	3F	3F	E4	A3	3F	3F	3F	3F	:
3:	03	13	23	33	43	53	63	73	3F	3F	E7	3F	3F	3F	3F	3F	:
4:	04	14	24	34	44	54	64	74	3F	3F	E9	A8	3F	3F	3F	3F	:
5:	05	15	25	35	45	55	65	75	3F	3F	E8	5E	3F	3F	3F	3F	:
6:	06	16	26	36	46	56	66	76	3F	3F	EA	3F	3F	3F	3F	3F	:
7:	07	17	27	37	47	57	67	77	3F	3F	EB	3F	3F	3F	3F	3F	:
8:	08	18	28	38	48	58	68	78	3F	3F	EE	3F	3F	3F	3F	3F	:
9:	09	19	29	39	49	59	69	79	3F	3F	EF	3F	3F	3F	3F	3F	:
A:	0A	1A	2A	3A	4A	5A	6A	7A	3F	3F	F4	3F	3F	3F	3F	3F	:
B:	0B	1B	2B	3B	4B	5B	6B	7B	3F	3F	F6	3F	3F	3F	3F	3F	:
C:	0C	1C	2C	3C	4C	5C	6C	7C	3F	3F	F9	3F	3F	3F	3F	3F	:
D:	0D	1D	2D	3D	4D	5D	6D	7D	3F	3F	FB	3F	3F	3F	3F	3F	:
E:	0E	1E	2E	3E	4E	5E	6E	7E	3F	3F	FC	3F	3F	3F	3F	3F	:
F:	0F	1F	2F	3F	4F	5F	6F	7F	3F	3F	3F	3F	3F	3F	3F	3F	:

Glossary

ACK (Acknowledgement, 06h) An affirmative response to a normal selection (indicating Ready to Receive) or a transmission (indicating message accepted).

AD1, AD2 (Address 1, Address 2) A two character address established as the address of a terminal. These characters are used to address a terminal in polling or selection or in the message heading. These characters identify the terminal from which a message is transmitted. On receipt of a message, the receiving station may use AD1 - AD2 to verify that the message originated at the polled terminal. AD1 and AD2 are represented by any characters from columns 2,3,4,5,6,7 of the ASCII code chart on page B-2, except the character DEL, column 7, row 15, shown as 7/15.

Application Status Block (ASB) Used by the High-Level Interface process to communicate the status of the reads and writes issued by an application. (Table 6-1 shows this block's format.)

ASB Application Status Block.

BCC (Block Check Character) A redundant character added to the end of a message for error detection and control. BCC is formed by taking a binary sum without carry on each of the 7 bits of the transmitted characters following SOH, including ETX, but excluding any SYN characters. The correct value of the character parity bit of the BCC is that which makes the sense of character parity the same as for text characters. BCC immediately follows ETX.

BMULTI (Burroughs Multipoint Communications Service) 1) A BTOS system service that provides access to a Burroughs Multipoint line or Poll/Select line to an application program . 2) The software product of which this service is a part.

BSL (Broadcast Select, 74h) a character used to indicate a broadcast message to all stations. In the broadcast sequence, AD1 - AD2 identifies the station that acknowledges receipt of the message. Broadcast select is followed immediately by a message without requiring acknowledgment of the selection.

BTE Burroughs Terminal Emulator.

BTOS A multiprogramming, message-passing operating system for B20/B25/XE520 family of devices.

CON (Contention, 07h) A character used to instruct all terminals that receive the instruction to go to the contention mode. NUL characters replace AD1 - AD2 in the contention sequence. There is no acknowledgment of the contention instruction.

Configuration A process of entering specific information (in the form of parameter values) about your hardware, software, and network arrangement to BMULTI or another program so that the program operates correctly on your system.

Context Manager A BTOS utility that enables several applications to run simultaneously on a BTOS workstation.

Control Station a station on a data link that is responsible for polling, selecting, and ensuring the orderly operation of that link. (Usually a control station is a large Burroughs computer.) The control station is responsible for initiating recovery procedures during abnormal conditions on the link. All stations on a multipoint network, other than the control station, are called terminal stations. These are usually terminals but may be microcomputers (such as BTOS workstations) or minicomputers.

Dc Interface The original procedural interface to BMULTI, used by applications wanting to access BMULTI services. It is limited to a single address per task.

DTS Data Transfer Service.

ENQ (Inquiry, 05h) A reply request control character. ENQ is used as the final character of a poll or of a select, when a response is required from the other station.

EOT (End of Transmission, 04h) A character transmitted by a terminal as a no traffic response to a poll. Receipt of EOT places the terminal in a control state listening for a polling or selection sequence. EOT may be transmitted instead of ETX to abort a transmission.

ETX (End of Text, 03h) A character used to indicate the end of a stream of characters identified as a text.

FSL (Fast Select, 73h) A character used to indicate a Fast select in a selection sequence transmitted by the central computer. Fast select is followed immediately by a message without requiring acknowledgment of the selection.

GSL (Group Select) A character used to indicate a message for a group of stations. In the group select sequence, AD1 - AD2 identifies the station that is to acknowledge receipt of the message. Group select is followed immediately by a message without requiring acknowledgment of the selection. Group select may be represented by any agreed on character selected from columns 2 through 6.

HLI (High-Level Interface) A procedural interface to BMULTI, intended primarily for COBOL application programmers who do not want to concentrate on the details of the BMULTI interface.

LLI (Low-Level Interface) A procedural interface to BMULTI, used by applications wanting to access BMULTI services.

Mark A symbol that indicates the beginning or ending of a set of data, such as a record.

Mp Interface A procedural interface to BMULTI, used by applications wanting to access BMULTI services. It allows up to three addresses per task.

MTI (Multiple Task Interface). See Mp Interface.

NAK (Negative Acknowledgement, 15h) A negative response to a selection (indicating Not Ready to Receive) or a transmission (indicating character parity failure for any character in a message or a failure of the BCC).

Parameters Variable values that are entered to customize a program or software for your system's specific requirements.

POL (Poll, 70h) Indicates a poll preceding ENQ in a polling sequence.

Ring buffer A buffer in which the newest information overwrites the oldest, i.e., information does not "overflow," but rather replaces the information at the beginning of the buffer.

RVI (Reverse Interrupt (DLE<) 103Ch) Sent by the control station in lieu of a positive acknowledgment (ACK) when the control station has priority messages to deliver. RVI is normally used in a group poll environment to request premature termination of a series of message transmissions, to enable the control station to either transmit return messages or to poll other terminals. On receipt of an RVI, the terminal should send EOT as soon as possible.

SEL (Select, 71h) Used to indicate a normal select preceding ENQ in a selection sequence.

SOH (Start of Heading, 01h) The first of a sequence of characters that form the heading. The heading also contains a terminal identification (AD1, AD2) and may contain transmission numbers (XMno). A heading is ended by STX.

STI (Single Task Interface) See Dc Interface.

STX (Start of Text, 02h) Precedes a sequence of characters that form the text of the transmission. STX terminates a heading.

SYN (Synchronous Idle, 16h) Used only with synchronous transmission in the absence of any other character to provide a signal for establishing and retaining synchronism. On initiating a synchronous transmission, several SYN characters are transmitted prior to the transmission of any character. This enables the receiving station to acquire character synchronization. SYN is also used as a time fill when no other characters are available for transmission at any point in a character sequence, except between ETX and the next following BCC. SYN is purged at the receiving station and is not included in the summation for BCC.

Transmission Number See XMno.

VAD Virtual Address.

Virtual Addresses (VADs) The method by which BMULTI avoids timing out inactive stations on the host computer. VADs act like active address in the idle state.

XMno (Transmission Number) A number identifying, in sequence, transmissions from or a transmission to a terminal. It is used optionally as part of a message header to assist in message recovery. Separate sets of transmission numbers are to be used for broadcast and group addressed messages.

Index

A

abort, 7-4
ACK (acknowledgment) character, 8-3 to 8-18
AD1, AD2 (Address 1, Address 2), 8-3 to 8-18
address:
  group poll, 3-1
  purging, 4-4
  state, 6-2
  station, 7-3
  used in OpenBMULTI, 9-3
application:
  examples, A-1
  interface with BMULTI, 6-1, 9-1 to 9-8, 10-1 to 10-11
  linking with BMULTI.lib, 6-2
  program, 6-1
ASCII (chart), B-1
asynchronous data communication, 8-1

B

BASIC program, 6-2, 9-1, A-1, D-1
baud rate, 3-3
BCC (Block Check Character), 7-15, 7-17, 8-4 to 8-18
BmCommand, 10-1
BmGetStatus, 10-6
BmOpenII, F-2
BmReport, 10-4
BmReportTimeout, 10-1
BmSetXlatMode, 10-10
BMULTI concepts, 6-1
BMULTI.lib, 6-2
BMULTI state machine, 7-1
BMULTI states:
  idle, 7-6
  offline, 7-6
  receive ready state, 7-15
  receiving and transmit ready state, 7-17
  receiving state, 7-14
  transmit ready, 7-8
  transmitting, 7-10
  transmitting and receive ready state, 7-12
  transmit and receive ready state, 7-13
B-Net support, 10-1, F-1
Broadcast Select (BSL) character, 7-3, 3-5, 8-13
BSL (Broadcast Select) character, 8-4 to 8-18
BTOS request codes for BMULTI, E-1

buffer:
general, 10-2, G-1 to G-3
receive, 7-4, F-1
transmit, 7-4, F-4

C

channel, 3-1, 3-2, F-2
check primitive, 6-3
CloseBMULTI, 9-3
COBOL:
configuring, D-1
general, 6-1, 6-2, 9-1, A-1, D-1
program, A-1
commands, 6-2
common information buffer format, 10-7
configuration files from BMULTI 5.0 and earlier, 3-4
configuration parameters, 3-2
configuration worksheet, 3-1
configurator, 3-5
configuring BMULTI, 3-1
CON (contention) character, 8-17, 8-18

D

data accountability, 8-2
DCA box (Unisys TDI/concantenation adapter), C-1
deinstalling BMULTI, 2-2

E

earlier BMULTI versions, F-2
end session command, 7-3, 7-5, 7-6, 9-3, 10-2
ENQ (inquiry) character, 8-1 to 8-18
EOT (end of transmission) character, 8-1 to 8-18
ETX (end of text) character, 8-1 to 8-18

F

fast select, 1-1, 7-5, 8-1 to 8-18
features, 1-1
files on the installation disk, 2-3, 2-4
FORTRAN, A-1, D-1
FSL (fast select) character, 8-1 to 8-18

G

group poll address, 1-1, 3-2, 8-6
group select, 1-1, 3-2, 8-15
GSL (group select) character, 3-2, 8-1 to 8-18

H

hardware requirements, 1-2, C-1
high-level procedural interface:
CloseBMULTI, 9-3,
general, 9-1
OpenBMULTI, 9-3
ReadBMULTI, 9-4
ResetBMULTI, 9-5
SelectBMULTI, 9-7
SetOptionBMULTI, 9-5
SetXlatModeBMULTI, 9-8
WriteBMULTI, 9-6
host:
host/NDL requirements, H-1
protocol, 8-1 to 8-18
system modem, C-1
timeout, 1-1, 10-1

I

idle, 7-4
idle line, 8-3
installation:
automatically installing BMULTI, 2-1
general, 2-1
hard disk systems installation, 2-1
interface:
general, 1-3, 6-1, G-1, H-1
high level, 9-1 to 9-8
low-level, 10-1 to 10-11

L

language configuration, A-1, D-1
line monitor, 5-1, 5-10
low-level interface:
BmCommand, 10-1
BmGetStatus, 10-8
BmOpenII, 10-3
BmReport, 10-4
BmReportTimeout, 10-1
BmSetXlatMode, 10-10
common information buffer format, 10-7
general , 10-1
station record format, 10-8

M

memory requirements, 1-2
modem, 3-3, C-1 (*see also* DCA Box)
modulus, 3-3
multipoint contention, 8-17

N

NAK (negative acknowledge) character, 7-3, 8-1 to 8-18

O

offline command, 7-3
online command, 7-3
OpenBMULTI, 9-3
operating system requirements, 1-2

P

Pascal program, 6-1, 9-1, A-1
POL (poll) character, 8-1 to 8-18
poll, 1-1, 7-3, 7-4, H-1 (*see also* group poll)
previous versions of BMULTI, 1-3
programs (samples), A-1
programming language configuration (*see* language configuration)
protocol description, 8-1 to 8-18
protocol handler, 7-13, 7-14
purge address, 4-4

R

ReadBmulti, 9-4
receive:
  buffer overflow, F-1
  command, 9-4
  delay, 3-3
  done, 7-15, 7-17, 10-4
  general, 3-3, 7-4, 8-1 to 8-18, 9-4
  ready, 7-6, 7-8, 7-10, 7-12, 7-13, 7-15
  state, 7-14
references, vii
report queue, 7-5
reports, 6-2
requests:
  codes, E-1
  general, 6-2
reset fast ready, 7-3, 7-5, 10-2
ResetBmulti, 9-5
response timeout, 8-3

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

No Postage
necessary
if mailed in the
United States

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA 19422

Postage Will Be Paid By Addressee

Unisys Corporation
ATTN: Corporate Product Information
Room C1 - NE19
P.O. Box 500
Blue Bell, PA 19422-9945 USA

No Postage
necessary
if mailed in the
United States

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA 19422

Postage Will Be Paid By Addressee

Unisys Corporation
ATTN: Corporate Product Information
Room C1 - NE19
P.O. Box 500
Blue Bell, PA 19422-9945 USA