OpenVMS VAX Card Reader, Line Printer, and LPA11–K I/O User's Reference Manual

Order Number: AA–PVXGA–TE

May 1993

This document contains the information necessary to interface directly with three device drivers (card reader driver, laboratory peripheral accelerator driver, and line printer driver) that are supplied as part of the OpenVMS VAX operating system. Several examples of programming techniques are included.

Revision/Update Information: This is a new manual.

Software Version: OpenVMS VAX Version 6.0

Digital Equipment Corporation
Maynard, Massachusetts

May 1993

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK6269

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

3 Line Printer Driver

A I/O Function Codes

Index

Examples

Figures

Tables

Preface

Intended Audience

This manual is intended for system programmers who want to take advantage of the time and space savings that result from direct use of I/O drivers. If you do not require such detailed knowledge of I/O drivers, use the device-independent services described in the *OpenVMS Record Management Services Reference Manual*.

Document Structure

This manual is organized into three chapters and one appendix, as follows:

- Chapter 1 discusses the card reader driver.
- Chapter 2 discusses the LPA11-K driver.
- Chapter 3 discusses the line printer drivers.
- Appendix A summarizes the QIO function codes, arguments, and function modifiers that these drivers use.

Associated Documents

The following documents provide additional information:

- *OpenVMS I/O User's Reference Manual*
- *OpenVMS System Services Reference Manual*
- *OpenVMS Programming Environment Manual*
- *OpenVMS Record Management Services Reference Manual*
- *VMS Device Support Manual*
- *LPA11-K Laboratory Peripheral Accelerator User's Guide*
- OpenVMS system messages documentation

Conventions

In this manual, every use of OpenVMS VAX means the OpenVMS VAX operating system.

The following conventions are also used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.

PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in a VMS file specification, or in the syntax of a substring specification in an assignment statement.)
{ }	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in online versions of the manual.
italic text	Italic text emphasizes important information, indicates variables, and indicates complete titles of manuals. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
-	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal, unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1

Card Reader Driver

This chapter describes the use of the card reader driver that supports the CR11 card reader.

1.1 Supported Card Reader Device

The CR11 card reader reads standard 80-column punched data cards.

1.2 Driver Features

The card reader driver provides the following features:

- Support for multiple controllers of the same type; for example, more than one CR11 can be used on the system
- Binary, packed Hollerith, and translated 026 or 029 read modes
- Unsolicited interrupt support for automatic card reader input spooling
- Special card punch combinations to indicate an end-of-file condition and to set the translation mode
- Error recovery

The following sections describe the read modes, special card punch combinations, and error recovery in greater detail.

The operating system provides the following card reader device- or function-dependent modifier bits for read data operations:

- IO$M_PACKED—Read packed Hollerith code
- IO$M_BINARY—Read binary code

If IO$M_PACKED is set, the data is packed and stored in sequential bytes of the input buffer. If IO$M_BINARY is set, the data is read and stored in sequential words of the input buffer. IO$M_BINARY takes precedence over IO$M_PACKED.

The read mode can also be set by a special card punch combination that sets the translation mode (see Section 1.2.1.2), or by the set mode function (see Section 1.4.3).

1.2.1 Special Card Punch Combinations

The card reader driver recognizes three special card punch combinations in column 1 of a card. One combination signals an end-of-file condition. The other two combinations set the current translation mode.

1.2.1.1 End-of-File Condition

A card with the 12-11-0-1-6-7-8-9 holes punched in column 1 signals an end-of-file condition. If the read mode is binary, the first eight columns must contain that punch combination.

1.2.1.2 Set Translation Mode

If the read mode is nonbinary, nonpacked Hollerith (the IO$M_BINARY and IO$M_PACKED function modifiers are not set), the current translation mode can be set to the 026 or 029 punch code. (Table 1–5 lists the 026 and 029 punch codes.) A card with the 12-2-4-8 holes punched in column 1 sets the translation mode to the 026 code. A card with the 12-0-2-4-6-8 holes punched in column 1 sets the translation mode to the 029 code. The translation mode can be changed as often as required.

If a translation mode card contains punched information in columns 2 through 80, it is ignored.

The system can read cards that were punched on an 026 punch or an 029 punch. By default, the translation mode is 029; that is, the system reads cards from an 029 punch. However, you can change the translation mode by using the following:

- The SET CARD_READER command
- Translation mode cards

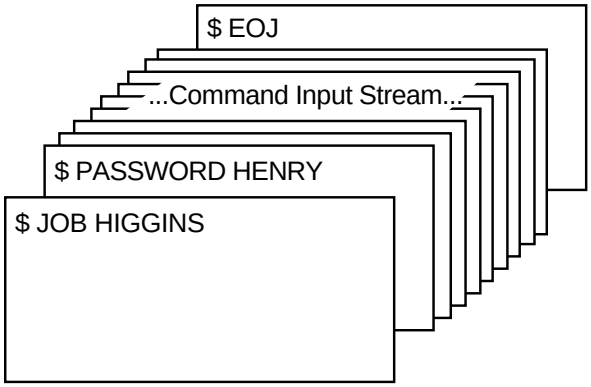
Use the SET CARD_READER command, with the /026 or /029 qualifier, to set the card reader to accept cards from either an 026 or an 029 card punch.

Logical, virtual, and physical read functions result in only one card being read. If a translation mode card is read, the read function is not completed, and another card is read immediately.

1.2.2 Submitting Batch Jobs Through the Card Reader

When you submit a batch job through a system card reader, precede the card deck containing the command procedure with cards containing JOB and PASSWORD commands. These cards specify your user name and password and, when executed, effect a login for you. The last card in the deck must contain the End of Job (EOJ) command. The EOJ card is equivalent to logging out. You can also use an overpunch card instead of an EOJ card to signal the end of a job. To do this, use an EOF card (12-11-0-1-6-7-8-9) overpunch or use the EOJ command. Figure 1–1 illustrates a card reader batch job.

Figure 1–1 A Card Reader Batch Job



ZK–0812–GE

When the system reads a job from the card reader, it validates the user name and password specified on the JOB and PASSWORD cards. Then, it copies the entire card deck into a temporary disk file named INPBATCH.COM in your default disk and directory, and it queues the job for batch execution. Thereafter, processing is the same as for jobs submitted interactively with the SUBMIT command. When the batch job is completed, the operating system deletes the INPBATCH.COM file.

You can prevent other users from seeing your password by suppressing printing when you keypunch the PASSWORD card.

1.2.3 Passing Data to Commands and Images

To pass data to commands and images in batch jobs that you submit through a card reader, you can do the following:

- Include the data in the command procedure by placing the data on the lines after the command or image that uses the data. Use the DECK and EOD commands if the data lines begin with dollar signs.
- Temporarily redefine SYS$INPUT as a file by using the DEFINE/USER_MODE command.

1.2.4 Error Recovery

The card reader driver performs the following error recovery operations:

- If the card reader is off-line for 30 seconds, a “device not ready” message is sent to the system operator.
- If a recoverable card reader failure is detected, a “device not ready” message is sent every 30 seconds to the system operator.
- The current operation is retried every two seconds to test for a changed situation, such as the removal of an error condition.
- The current I/O operation can be canceled at the next timeout without the card reader being on line. When the card reader comes on line, device operation resumes automatically.

When a recoverable card reader failure is detected and an error message is displayed on the system operator console, examine the card reader indicator lights to determine the reason for the failure. Any errors that occur must be fixed manually. The recovery is transparent to the user program issuing the I/O request.

The four categories of card reader failures and their respective recovery procedures are as follows:

- Pick check—The next card cannot be delivered from the input hopper to the read mechanism. To recover from this error, remove the next card to be read from the input hopper and smooth the leading edge (the edge that enters the read mechanism first). Replace the card in the input hopper and press the RESET button. The card reader operation resumes automatically. If a pick check error occurs again on the same card, remove the card from the input hopper and repunch it. Place the duplicate card in the input hopper and press the RESET button. If the problem persists, either an adjustment is required, or nonstandard cards are in the input hopper.
- Stack check—The card just read did not stack properly in the output hopper. To recover from this error, remove the last card read from the output hopper and examine it. If it is excessively worn or mutilated, repunch it. Place either card in the read station of the input hopper and press the RESET button. The card reader operation resumes automatically. If the stack check error recurs immediately, an adjustment is required.
- Hopper check—Either the input hopper is empty or the output hopper is full. To recover from this error, examine the input hopper and, if empty, either load the next deck of input cards or an end-of-file card. If the input hopper is not empty, remove the cards that have accumulated in the output hopper and press the RESET button. The card reader operation resumes automatically.
- Read check—The last card was read incorrectly. To recover from this error, remove the last card from the output hopper and examine it. If it is excessively worn, mutilated, or contains punches before column 0 or after column 80, repunch the card. Place either card in the read station of the input hopper and press the RESET button. The card reader operation resumes automatically. If the read check error recurs immediately, an adjustment is necessary.

1.3 Card Reader Driver Device Information

You can obtain information on card reader characteristics by using the Get Device/Volume Information ($GETDVI) system service. See the *OpenVMS System Services Reference Manual*.

$GETDVI returns card reader characteristics when you specify the item codes DVI$_DEVCHAR and DVI$_DEVDEPEND. Tables 1–1 and 1–2 list these characteristics. The $DEVDEF macro defines the device-independent characteristics; the $CRDEF macro defines the device-dependent characteristics.

DVI$_DEVTYPE and DVI$_DEVCLASS return the device type and device class names, which are defined by the $DCDEF macro. The device class for card readers is DC$_CARD. The device type for the CR11 is DT$_CR11. DVI$_DEVBUFSIZ returns the buffer size. The default buffer size to be used for all card reader devices is 80 bytes.

Table 1–1 Card Reader Device-Independent Characteristics

Characteristic[1]	Meaning
	Dynamic Bit (Conditionally Set)
DEV$M_AVL	Device is on line and available
	Static Bits (Always Set)
DEV$M_IDV	Device is capable of input
DEV$M_REC	Device is record-oriented

[1]Defined by the $DEVDEF macro.

Table 1–2 Device-Dependent Characteristics for Card Readers

Value[1]	Meaning	
CR$V_TMODE CR$S_TMODE	Specifies the translation mode for nonbinary, nonpacked Hollerith data transfers.[2] Possible values are:	
	CR$K_T026	Translate according to 026 punch code
	CR$K_T029	Translate according to 029 punch code

[1]Defined by the $CRDEF macro.

[2]Section 1.2.1.2 describes the set translation mode punch code.

1.4 Card Reader Function Codes

The card reader driver can perform logical, virtual, and physical I/O functions. Table 1–3 lists these functions and their function codes. These functions are described in more detail in the sections that follow.

Table 1–3 Card Reader I/O Functions

Function Code and Arguments	Type[1]	Function Modifiers	Function
IO$_READLBLK P1,P2	L	IO$M_BINARY IO$M_PACKED	Read logical block.
IO$_READVBLK P1,P2	V	IO$M_BINARY IO$M_PACKED	Read virtual block.
IO$_READPBLK P1,P2	P	IO$M_BINARY IO$M_PACKED	Read physical block.
IO$_SENSEMODE	L		Sense the card reader characteristics and return them in the I/O status block.
IO$_SETMODE P1	L		Set card reader characteristics for subsequent operations.

[1]V = virtual; L = logical; P = physical

(continued on next page)

Table 1–3 (Cont.) Card Reader I/O Functions

Function Code and Arguments	Type[1]	Function Modifiers	Function
IO$_SETCHAR P1	P		Set card reader characteristics for subsequent operations.

[1]V = virtual; L = logical; P = physical

1.4.1 Read

Read is a function that reads data from the next card in the card reader input hopper into the designated memory buffer in the specified format. Only one card is read each time a read function is specified.

The operating system provides the following read function codes:

- IO$_READVBLK—Read virtual block
- IO$_READLBLK—Read logical block
- IO$_READPBLK—Read physical block

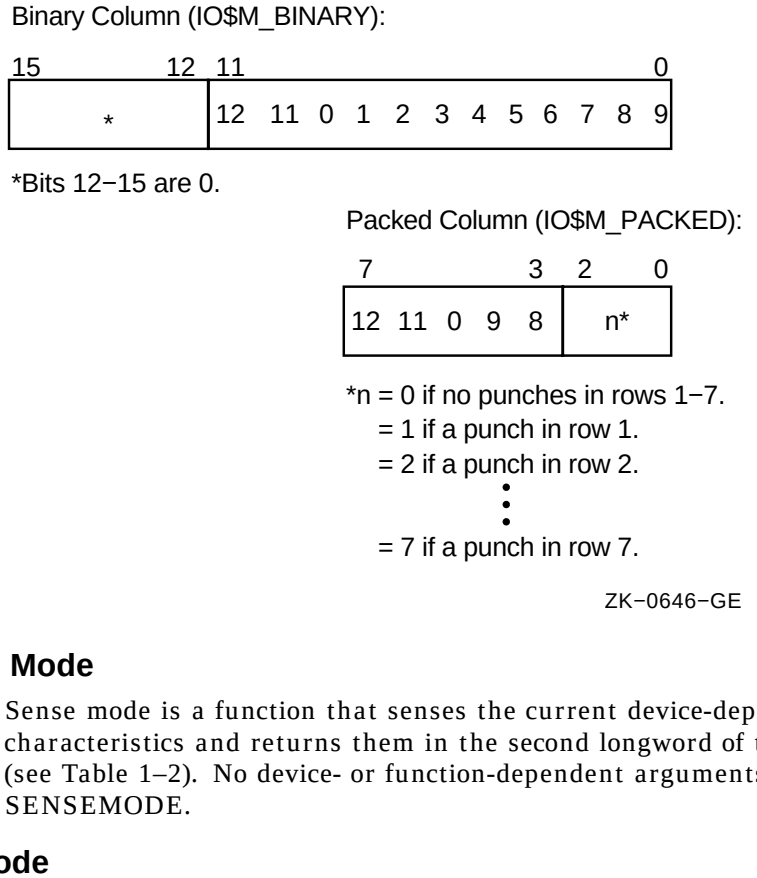
The following function-dependent arguments are used with these codes:

- P1—The starting virtual address of the buffer that is to receive the data
- P2—The number of bytes that are to be read in the specified format

The read binary function modifier (IO$M_BINARY) and the read packed Hollerith function modifier (IO$M_PACKED) can be used with all read functions. If IO$M_BINARY is specified, successive columns of data are stored in sequential word locations of the input buffer. If IO$M_PACKED is specified, successive columns of data are packed and stored in sequential byte locations of the input buffer. If neither of these function modifiers is specified, successive columns of data are translated in the current mode (026 or 029) and are stored in sequential bytes of the input buffer. Figure 1–2 shows how data is stored by IO$M_BINARY and IO$M_PACKED.

Regardless of the byte count specified by the P2 argument, a maximum of 160 bytes of data for binary read operations and 80 bytes of data for nonbinary read operations (IO$M_PACKED, or 026 or 029 modes) are transferred to the input buffer. If P2 specifies less than the maximum quantity for the respective mode, only the number of bytes specified are transferred; any remaining buffer locations are not filled with data.

Figure 1–2 Binary and Packed Column Storage



1.4.2 Sense Mode

Sense mode is a function that senses the current device-dependent card reader characteristics and returns them in the second longword of the I/O status block (see Table 1–2). No device- or function-dependent arguments are used with IO$_SENSEMODE.

1.4.3 Set Mode

Set mode operations affect the operation and characteristics of the associated card reader device. The operating system defines the following types of set mode functions:

- Set mode
- Set characteristic

1.4.3.1 Set Mode

The set mode function affects the characteristics of the associated card reader. Set mode is a logical I/O function and requires the access privilege necessary to perform logical I/O. The following function code is provided.

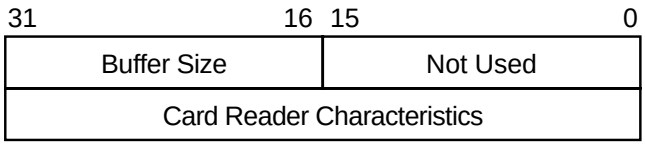
- IO$_SETMODE

This function takes the following device- or function-dependent argument:

- P1—The address of a characteristics buffer

Figure 1–3 shows the quadword set mode characteristics buffer.

Figure 1–3 Set Mode Characteristics Buffer



ZK–0647–GE

Table 1–4 lists the card reader characteristics and their meanings. The $CRDEF macro defines the characteristics values. Table 1–5 lists the 026 and 029 card reader codes.

Table 1–4 Set Mode and Set Characteristic Card Reader Characteristics

Value[1]	Meaning	
CR$V_TMODE CR$S_TMODE	Specifies the translation mode for nonbinary, nonpacked Hollerith data transfers. Possible values are:	
	CR$K_T026	Translate according to 026 punch code
	CR$K_T029	Translate according to 029 punch code

[1] If neither the 026 nor 029 mode is specified, the default mode can be set by the SET CARD_READER command.

Table 1–5 Card Reader Codes

Character	$ASCII_8$	DEC029	DEC026
{	173	12 0	12 0
}	175	11 0	11 0
SPACE	40	NONE	NONE
!	41	11 8 2	12 8 7
"	42	8 7	0 8 5
_	43	8 3	0 8 6
$	44	11 8 3	11 8 3
%	45	0 8 4	0 8 7
&	46	12	11 8 7
′	47	8 5	8 6
(	50	12 8 5	0 8 4
)	51	11 8 5	12 8 4
*	52	11 8 4	11 8 4
+	53	12 8 6	12

(continued on next page)

Table 1–5 (Cont.) Card Reader Codes

Character	$ASCII_8$	DEC029	DEC026
,	54	0 8 3	0 8 3
-	55	11	11
.	56	12 8 3	12 8 3
/	57	0 1	0 1
0	60	0	0
1	61	1	1
2	62	2	2
3	63	3	3
4	64	4	4
5	65	5	5
6	66	6	6
7	67	7	7
8	70	8	8
9	71	9	9
:	72	8 2	11 8 2
;	73	11 8 6	0 8 2
<	74	12 8 4	12 8 6
=	75	8 6	8 3
>	76	0 8 6	11 8 6
?	77	0 8 7	12 8 2
@	100	8 4	8 4
A	101	12 1	12 1
B	102	12 2	12 2
C	103	12 3	12 3
D	104	12 4	12 4
E	105	12 5	12 5
F	106	12 6	12 6
G	107	12 7	12 7
H	110	12 8	12 8
I	111	12 9	12 9
J	112	11 1	11 1
K	113	11 2	11 2
L	114	11 3	11 3
M	115	11 4	11 4
N	116	11 5	11 5
O	117	11 6	11 6
P	120	11 7	11 7
Q	121	11 8	11 8

(continued on next page)

Table 1–5 (Cont.) Card Reader Codes

Character	$ASCII_8$	DEC029	DEC026
R	122	11 9	11 9
S	123	0 2	0 2
T	124	0 3	0 3
U	125	0 4	0 4
V	126	0 5	0 5
W	127	0 6	0 6
X	130	0 7	0 7
Y	131	0 8	0 8
Z	132	0 9	0 9
[	133	12 8 2	11 8 5
\	134	11 8 7	8 7
]	135	0 8 2	12 8 5
↑ or ^	136	12 8 7	8 5
← or _	137	0 8 5	8 2

Application programs that change specific card reader characteristics should first use the IO$_SENSEMODE function to read the current characteristics, modify them, and then use the set mode function to write back the results. Failure to follow this sequence results in clearing any previously set characteristic.

1.4.3.2 Set Characteristic

The set characteristic function also affects the characteristics of the associated card reader device. Set characteristic is a physical I/O function, and requires the access privilege necessary to perform physical I/O functions. The following function code is provided:

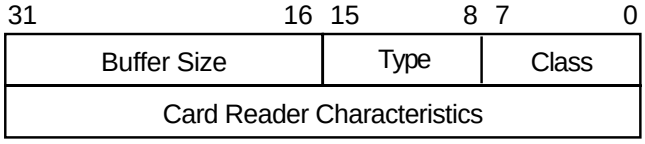
- IO$_SETCHAR

This function takes the following device- or function-dependent argument:

- P1—The address of a characteristics buffer

Figure 1–4 shows the set characteristic characteristics buffer.

Figure 1–4 Set Characteristic Buffer



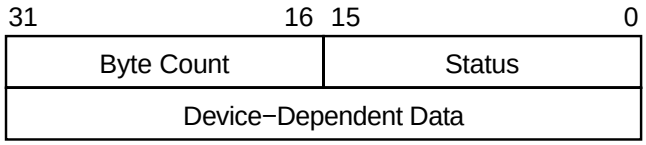
ZK–0648–GE

The device type value is DT$_CR11. The device class value is DC$_CARD. Table 1–4 lists the card reader characteristics for the Set Characteristic function.

1.5 I/O Status Block

The I/O status block (IOSB) format for QIO functions on the card reader is shown in Figure 1–5. Appendix A lists the status returns for these functions. (The OpenVMS system messages documentation provides explanations and suggested user actions for these returns.) Table 1–2 lists the device-dependent data returned in the second longword. The IO$_SENSEMODE function can be used to obtain this data.

Figure 1–5 IOSB Contents



ZK−0649−GE

2

Laboratory Peripheral Accelerator Driver

This chapter describes the laboratory peripheral accelerator (LPA11-K) driver and the high-level language procedure library that interfaces with it. The procedure library is implemented with callable assembly language routines that translate arguments into the format required by the LPA11-K driver and that handle buffer chaining operations. Routines for loading the microcode and initializing the device are also described.

Refer to the *LPA11-K Laboratory Peripheral Accelerator User's Guide* for additional information.

2.1 Supported Device

The LPA11-K is a peripheral device that controls analog-to-digital (A/D) and digital-to-analog (D/A) converters, digital I/O registers, and real-time clocks. It is connected to the VAX processor through the UNIBUS adapter.

The LPA11-K is a fast, flexible microprocessor subsystem designed for applications requiring high-speed, concurrent data acquisition and data reduction. The LPA11-K allows aggregate analog input and output rates of up to 150,000 samples per second. The maximum aggregate digital input and output rate is 15,000 samples per second.

Table 2–1 lists the useful minimum and maximum LPA11-K configurations supported by the operating system.

Table 2–1 Minimum and Maximum Configurations per LPA11-K

Minimum	Maximum
1 DD11-Cx or Dx backplane	2 DD11-Cx or Dx backplanes
1 KW11-K real-time clock	1 KW11-K real-time clock
1 of the following:	2 AD11-K A/D converters
AD11-K A/D converter	2 AM11-K multiplexers for AD11-K converters
AA11-K A/D converter	1 AA11-K D/A converter
DR11-K digital I/O register	5 DR11-K digital I/O registers

2.1.1 LPA11-K Modes of Operation

The LPA11-K operates in two modes: dedicated and multirequest.

In dedicated mode, only one user (one request), can be active at a time, and only analog I/O data transfers are supported. Up to two A/D converters can be controlled simultaneously. One D/A converter can be controlled at a time. Sampling is initiated either by an overflow of the real-time clock or by an externally supplied signal. Dedicated mode provides sampling rates of up to 150,000 samples per second.

In multirequest mode, sampling from all of the devices listed in Table 2–1 is supported. The LPA11-K operates like a multicontroller device; up to eight requests (from one through eight users) can be active simultaneously. The sampling rate for each user is a multiple of the common real-time clock rate. Independent rates can be maintained for each user. Both the sampling rate and the device type are specified as part of each data transfer request. Multirequest mode provides a maximum aggregate sampling rate of 15,000 samples per second.

2.1.2 Errors

The LPA11-K returns the following classes of errors:

1. Errors associated with the issuance of a new LPA11-K command (SS$_DEVCMDERR)
2. Errors associated with an active data transfer request (SS$_DEVREQERR)
3. Fatal hardware errors that affect all LPA11-K activity (SS$_CTRLERR)

The *LPA11-K Laboratory Peripheral Accelerator User's Guide* lists these three classes of errors and the specific error codes for each class. The LPA11-K aborts all active requests if any of the following conditions occur:

- Power failure
- Device timeout
- Fatal error

Power failure is reported to any active users when power is recovered.

The LADRIVER times out all $QIOs after two seconds if they have not completed. The driver does not provide any parameters that allow the user to change the length of the timeout.

The timeout period applied to all $QIOs can be changed with the following PATCH commands executed from a privileged account:

```
$ PATCH SYS$SYSTEM:LADRIVER.EXE/OUTPUT=SYS$SYSTEM:LADRIVER.EXE
PATCH>SET ECO 25
PATCH>REPLACE/INSTRUCTION LA$TIMEOUT_VALUE
OLD>'PUSHL     I^#00000002'
OLD>EXIT
NEW>'PUSHL     I^#0000003C'
NEW>EXIT
PATCH>UPDATE
PATCH>EXIT
```

Substitute the desired timeout value for the "0000003C" in the example above. When you reboot, the system loads the new copy of the driver containing the new timeout value.

Device timeouts are monitored only when a new command is issued. For data transfers, the time between buffer full interrupts is not defined. Thus, no timeout errors are reported on a buffer-to-buffer basis.

If a required resource is not available to a process, an error message is returned immediately. The driver does not place the process in the resource wait mode.

2.2 Supporting Software

The LPA11-K is supported by a device driver, a high-level language procedure library of support routines, and routines for loading the microcode and initializing the device. The system software and support routines provide a control path for synchronizing the use of buffers, specifying requests, and starting and stopping requests; the actual data algorithms for the laboratory data acquisition I/O devices are accomplished by the LPA11-K.

The LPA11-K driver and the associated I/O interface have the following features:

- They permit multiple LPA11-K subsystems on a single UNIBUS adapter.
- They operate as an integral part of the operating system.
- They can be loaded on a running system without relinking the executive.
- They handle I/O requests, function dispatching, UNIBUS adapter map allocation, interrupts, and error reporting for multiple LPA11-K subsystems.
- The LPA11-K functions as a multibuffered device. Up to eight buffer areas can be defined per request. Up to eight requests can be handled simultaneously. Buffer areas can be reused after the data they contain is processed.
- Because the LPA11-K chains buffer areas automatically, a start data transfer request can transfer an infinite and noninterrupted amount of data.
- Multiple ASTs are dynamically queued by the driver to indicate when a buffer has been filled (the data is available for processing) or emptied (the buffer is available for new data).

The high-level language support routines have the following features:

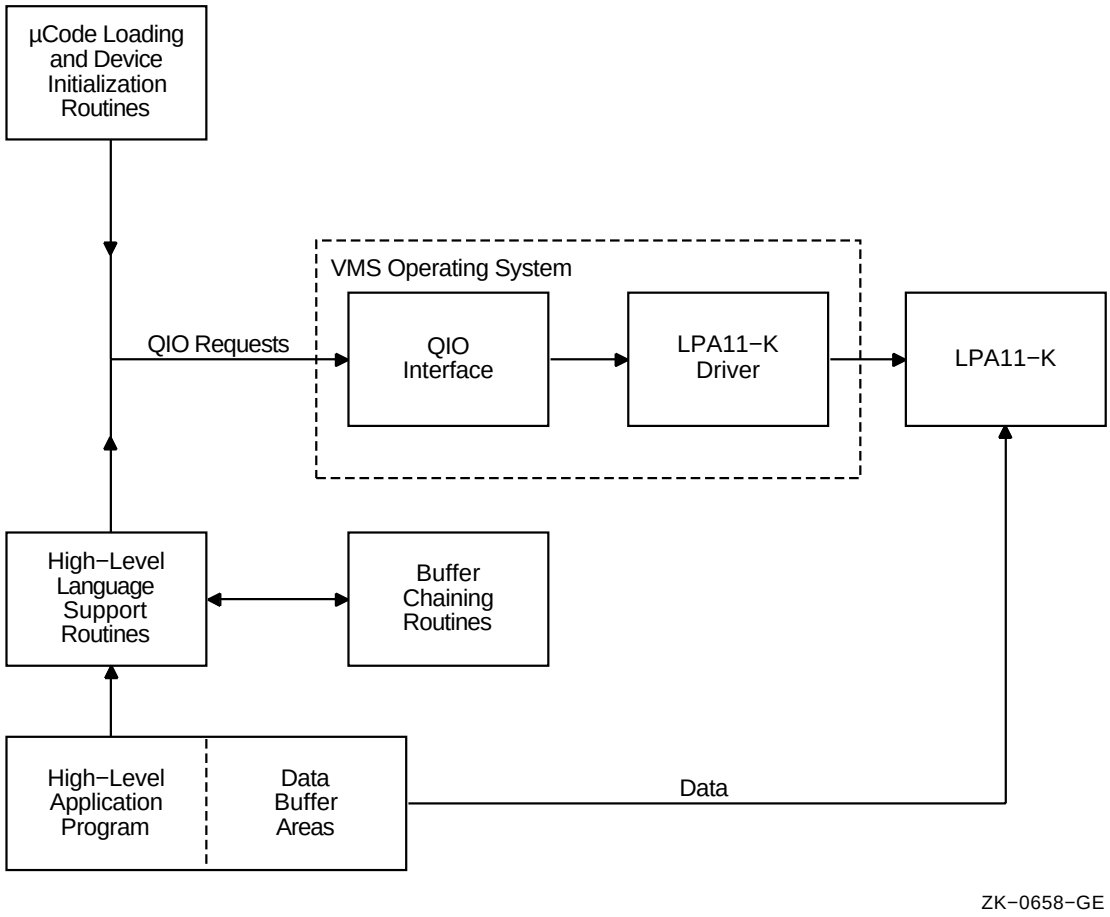
- They translate arguments provided in the high-level language calls into the format required for the Queue I/O interface.
- They provide a buffer chaining capability for a multibuffering environment by maintaining queues of used, in use, and available buffers.
- They adhere to all conventions for calling sequences, use of shareable resources, and reentrancy.
- They can be part of a resident global library, or they can be linked into a process image as needed.

The routines for loading microcode and initializing devices have the following features:

- They execute, as separate processes, images that issue I/O requests. These I/O requests initiate microcode image loading, start the LPA11-K subsystem, and automatically configure the peripheral devices on the LPA11-K internal I/O bus.
- They can be executed at the request of the user or an operator.
- They can be executed at the request of other processes.
- They can be executed automatically when the system is initialized and on power recovery.

Figure 2–1 shows the relationship of the supporting software to the LPA11-K.

Figure 2–1 Relationship of Supporting Software to LPA11-K



2.3 LPA11-K Device Information

You can obtain information on all peripheral data acquisition devices on the LPA11-K internal I/O bus by using the Get Volume Information ($GETDVI) system service. (See the *OpenVMS System Services Reference Manual*.)

$GETDVI returns device characteristics when you specify the item codes DVI$_DEVCHAR and DVI$_DEVDEPEND. Tables 2–2 and 2–3 list these characteristics. The $DEVDEF macro defines the device-independent characteristics; the $LADEF macro defines the device-dependent characteristics. Device-dependent characteristics are set by the set clock, initialize, and load microcode I/O functions to any one of, or a combination of, the values listed in Table 2–3.

DVI$_DEVCLASS and DVI$_DEVTYPE return the device class and device type names, which are defined by the $DCDEF macro. The device class for the LPA11-K is DC$_REALTIME; the device type is DT$_LPA11. DVI$_DEVBUFSIZ is not applicable to the LPA11-K.

Table 2–2 LPA11-K Device-Independent Characteristics

Characteristic[1]	Meaning
Dynamic Bit (Conditionally Set)	
DEV$M_AVL	Device is online and available.
Static Bits (Always Set)	
DEV$M_IDV	Device is capable of input.
DEV$M_ODV	Device is capable of output.
DEV$M_RTM	Device is real-time.
DEV$M_SHR	Device is shareable.

[1]Defined by the $DEVDEF macro.

Table 2–3 LPA11-K Device-Dependent Characteristics

Field[1]	Meaning
LA$M_MCVALID LA$V_MCVALID	The load microcode I/O function (IO$_LOADMCODE) was performed successfully. LA$M_MCVALID is set by IO$_ LOADMCODE. Each microword is verified by reading it back and comparing it with the specified value. LA$M_MCVALID is cleared if there is no match.
LA$V_MCTYPE LA$S_MCTYPE	The microcode type, set by the load microcode I/O function (IO$_ LOADMCODE), is one of the following values:

Value	Meaning
LA$K_MRMCODE	Microcode type is in multirequest mode.
LA$K_ADMCODE	Microcode type is in dedicated A/D mode.
LA$K_DAMCODE	Microcode type is in dedicated D/A mode.

[1]Defined by the $LADEF macro.

(continued on next page)

Table 2–3 (Cont.) LPA11-K Device-Dependent Characteristics

Field[1]	Meaning
LA$V_CONFIG LA$S_CONFIG	The bit positions, set by the initialize I/O function (IO$_INITIALIZE), for the peripheral data acquisition devices on the LPA11-K internal I/O bus are one or more of the following:

Value	Meaning
LA$V_CLOCKA LA$M_CLOCKA	Clock A
LA$V_CLOCKB LA$M_CLOCKB	Clock B
LA$V_AD1 LA$M_AD1	A/D device 1
LA$V_AD2 LA$M_AD2	A/D device 2
LA$V_DA LA$M_DA	D/A device 1
LA$V_DIO1 LA$M_DIO1	Digital I/O buffer 1
LA$V_DIO2 LA$M_DIO2	Digital I/O buffer 2
LA$V_DIO3 LA$M_DIO3	Digital I/O buffer 3
LA$V_DIO4 LA$M_DIO4	Digital I/O buffer 4
LA$V_DIO5 LA$M_DIO5	Digital I/O buffer 5

LA$V_RATE
LA$S_RATE — The Clock A rate, which is set by the set clock function (IO$_SETCLOCK), is one of the following values:

Value	Meaning
0	Stopped
1	1 MHz
2	100 kHz
3	10 kHz
4	1 kHz
5	100 Hz
6	Schmidt trigger
7	Line frequency

LA$V_PRESET
LA$S_PRESET — The Clock A preset value set by the set clock function (IO$_SETCLOCK). (The value is in two's complement form in the range 0 through 65,535.) The clock rate divided by the clock preset value yields the clock overflow rate.

[1]Defined by the $LADEF macro.

2.4 LPA11-K Function Codes

The LPA11-K I/O functions are as follows:

- Load the microcode into the LPA11-K.
- Start the LPA11-K microprocessor.
- Initialize the LPA11-K subsystem.
- Set the LPA11-K real-time clock rate.
- Start a data transfer request.

The first three functions are normally performed by the loader process, not by the user's data transfer program. See Section 2.5.21 for a description of the loader process interface.

The Cancel I/O on Channel ($CANCEL) system service is used to abort data transfers.

2.4.1 Load Microcode

This I/O function resets the LPA11-K and loads an image of LPA11-K microcode. Physical I/O privilege is required. The following function code is provided:

- IO$_LOADMCODE—Load microcode

The load microcode function takes the following device- or function-dependent arguments:

- P1—The starting virtual address of the microcode image that is to be loaded into the LPA11-K
- P2—The number of bytes (usually 2048) that are to be loaded
- P3—The starting microprogram address (usually 0) in the LPA11-K that is to receive the microcode

If any data transfer requests are active at the time a load microcode request is issued, the load request is rejected and SS$_DEVACTIVE is returned in the I/O status block.

Each microword is verified by comparing it with the specified value in memory. If all words match (the microcode was loaded successfully), the driver sets the microcode valid bit (LA$V_MCVALID) in the device-dependent characteristics longword (see Table 2–3). If there is no match, SS$_DATACHECK is returned in the I/O status block and LA$V_MCVALID is cleared to indicate that the microcode was not properly loaded. If the microcode was loaded successfully, the driver stores one of the microcode type values (LAK_MRCODE, LAK_ADCODE, or LA$K_DAMCODE) in the characteristics longword.

After a load microcode function is completed, the second word of the I/O status block contains the number of bytes loaded.

2.4.2 Start Microprocessor

This I/O function resets the LPA11-K and starts (or restarts) the LPA11-K microprocessor. Physical I/O privilege is required. The following function code is provided:

- IO$_STARTMPROC—Start microprocessor

This function code takes no device- or function-dependent arguments.

The start microprocessor function can return five error codes in the I/O status block (see Section 2.6):

SS$_CTRLERR SS$_DEVACTIVE SS$_MCNOTVALID
SS$_POWERFAIL SS$_TIMEOUT

The *LPA11-K Laboratory Peripheral Accelerator User's Guide* provides additional information on error codes.

2.4.3 Initialize LPA11-K

This I/O function issues a subsystem initialize command to the LPA11-K. This command specifies LPA11-K laboratory I/O device addresses and other table information for the subsystem. It is issued only once after restarting the subsystem and before any other LPA11-K command is given. Physical I/O privilege is required. The VMS operating system defines the following function code:

- IO$_INITIALIZE—Initialize LPA11-K

The initialize LPA11-K function takes the following device- or function-dependent arguments:

- P1—The starting, word-aligned, virtual address of the initialize command table in the user process. This table is read once by the LPA11-K during the execution of the initialize command. See the *LPA11-K Laboratory Peripheral Accelerator User's Guide* for additional information.
- P2—Length of the initialize command buffer (always 278 bytes).

If the initialize function is completed successfully, the appropriate device configuration values are set in the device-dependent characteristics longword (see Table 2–3).

The initialize function can return the following 10 error codes in the I/O status block:

SS$_BUFNOTALIGN SS$_CANCEL SS$_CTRLERR
SS$_DEVCMDERR SS$_INCLENGTH SS$_INSFMAPREG
SS$_IVMODE SS$_MCNOTVALID SS$_POWERFAIL
SS$_TIMEOUT

If a device specified in the initialize command table is not in the LPA11-K configuration, an error condition (SS$_DEVCMDERR) occurs and the address of the first device not found is returned in the LPA11-K maintenance status register (see Section 2.6). A program can use this characteristic to poll the LPA11-K and determine the current device configuration.

2.4.4 Set Clock

This virtual function issues a clock control command to the LPA11-K. The clock control command specifies information necessary to start, stop, or change the sample rate at which the real-time clock runs on the LPA11-K subsystem.

Note

If the LPA11-K has more than one user, caution should be exercised when the clock rate is changed. In multirequest mode, a change in the clock rate affects all users.

The following function code is provided:

- IO$_SETCLOCK—Set clock

The set clock function takes the following device- or function-dependent arguments:

- P2—Mode of operation. The operating system defines the following clock start mode word (hexadecimal) values:

Value	Meaning
1	KW11-K Clock A
11	KW11-K Clock B

- P3—Clock control and status. The operating system defines the following clock status word (hexadecimal) values:

Value	Meaning
0	Stop clock
143	1 MHz clock rate
145	100 kHz clock rate
147	10 kHz clock rate
149	1 kHz clock rate
14B	100 Hz clock rate
14D	Clock rate is Schmidt trigger 1
14F	Clock rate is line frequency

- P4—The two's complement of the real-time clock preset value. The range is 16 bits for the KW11-K Clock A and 8 bits for the KW11-K Clock B.

The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes the clock start mode word and the clock status word in greater detail.

If the set clock function is completed successfully for Clock A, the clock rate and preset values are stored in the device-dependent characteristics longword (see Table 2–3).

The set clock function can return six error codes in the I/O status block (see Section 2.6):

SS$_CANCEL	SS$_CTRLERR	SS$_DEVCMDERR
SS$_MCNOTVALID	SS$_POWERFAIL	SS$_TIMEOUT

The *LPA11-K Laboratory Peripheral Accelerator User's Guide* provides additional information on error codes.

2.4.5 Start Data Transfer Request

This virtual I/O function issues a data transfer start command that specifies the buffer addresses, sample mode, and sample parameters used by the LPA11-K. This information is passed to the data transfer command table. The following function code is provided:

- IO$_STARTDATA—Start data transfer request

The start data transfer request function takes the following function modifier:

- IO$M_SETEVF—Set event flag

The start data transfer request function takes the following device- or function-dependent arguments:

- P1—The starting virtual address of the data transfer command table in the user's process.
- P2—The length in bytes (always 40) of the data transfer command table.
- P3—The AST address of the normal buffer completion AST routine (optional).
- P4—The AST address of the buffer overrun completion AST routine (optional). This argument is used only when the buffer overrun bit (LA$M_BFROVRN) is set, that is, when a buffer overrun condition is classified as a nonfatal error.

A buffer overrun condition differs from a data overrun condition. The LPA11-K fetches data from, or stores data in, memory. If data cannot be fetched quickly enough (for example, when there is too much UNIBUS activity) a data underrun condition occurs. If data cannot be stored quickly enough, a data overrun condition occurs. After each buffer is filled or emptied, the LPA11-K obtains the index number of the next buffer to process from the user status word (USW). (See the *LPA11-K Laboratory Peripheral Accelerator User's Guide*.) A buffer overrun condition occurs if the LPA11-K fills or empties buffers faster than the application program can supply new buffers. For example, buffer overrun can occur when the sampling rate is too high, the buffers are too small, or the system load is too heavy.

The LPA11-K driver accesses the 10-longword data transfer command table (shown in Figure 2–2) when the data transfer start command is processed. After the command is accepted and data transfers begin, the driver does not access the table.

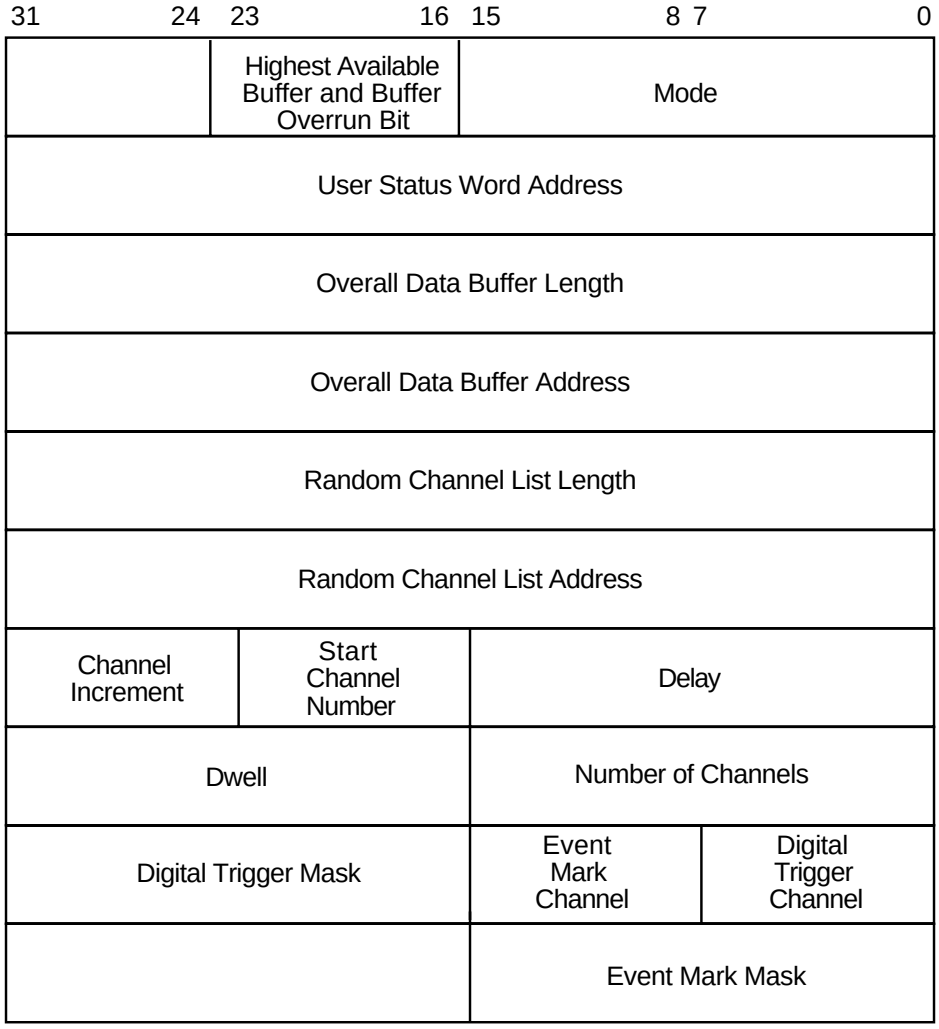
In the first longword of the data transfer command table, the first 2 bytes contain the LPA11-K start data transfer request mode word. (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes the functions of this word.)

The third byte contains the number (0–7) of the highest buffer available and the buffer overrun flag bit (bit 23; values: LA$M_BFROVRN and LA$V_BFROVRN). If this bit is set, a buffer overrun condition is a nonfatal error.

The second longword contains the user status word address (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*). This virtual address points to a 2-byte area in the user-process space and must be word aligned.

The third longword contains the size (in bytes) of the overall buffer area. The virtual address in the fourth longword is the beginning address of this area. This address must be longword aligned. The overall buffer area contains a specified number of buffers (the number of the highest available buffer specified in the first longword plus one). Individual buffers are subject to length restrictions: in multirequest mode the length must be in multiples of 2 bytes; in dedicated mode the length must be in multiples of 4 bytes. All data buffers are virtually contiguous for each data transfer request.

Figure 2–2 Data Transfer Command Table



ZK–0660–GE

The fifth and sixth longwords contain the random channel list (RCL) length and address, respectively. The RCL address must be word aligned. The last word in the RCL must have bit 15 set. (See the *LPA11-K Laboratory Peripheral Accelerator User's Guide* for additional information on the RCL.)

The seventh through tenth longwords contain LPA11-K-specific sample parameters. The driver passes these parameters directly to the LPA11-K. (See the *LPA11-K Laboratory Peripheral Accelerator User's Guide* for a detailed description of their functions.)

The start data transfer request function can return the following 15 error codes in the I/O status block (see Section 2.6):

SS$_ABORT	SS$_BUFNOTALIGN	SS$_CANCEL
SS$_CTRLERR	SS$_DEVCMDERR	SS$_DEVREQERR
SS$_EXQUOTA	SS$_INCLENGTH	SS$_INSFBUFDP
SS$_INSFMAPREG	SS$_INSFMEM	SS$_MCNOTVALID
SS$_PARITY	SS$_POWERFAIL	SS$_TIMEOUT

Data buffers are chained and reused as the LPA11-K and the user process dispose of the data. As each buffer is filled or emptied, the LPA11-K driver notifies the application process either by setting the event flag specified by the QIO request **efn** argument or by queuing an AST. Since buffer use is a continuing process, the event flag is set or the AST is queued a number of times. The user process must clear the event flag (or receive the AST), process the data, and specify the next buffer for the LPA11-K to use.

If the set event flag function modifier (IO$M_SETEVF) is specified, the event flag is set repeatedly: when the data transfer request is started, after each buffer completion, and when the request completes. If IO$M_SETEVF is not specified, the event flag is set only when the request completes.

ASTs are preferred over event flags for synchronizing a program with the LPA11-K, because AST delivery is a queued process, while the setting of event flags is not. If only event flags are used, buffer status may be lost.

Three AST addresses can be specified. For normal data buffer transactions the AST address specified in the P3 argument is used. If the buffer overrun bit in the data transfer command table is set and an overrun condition occurs, the AST address specified in the P4 argument is used. The AST address specified in the astadr argument of the QIO request is used when the entire data transfer request is completed. The **astprm** argument specified in the QIO request is passed to all three AST routines.

If insufficient dynamic memory is available to allocate an AST block, an error (SS$_INSFMEM) is returned. If the user does not have sufficient AST quota remaining to allocate an AST block, an error (SS$_EXQUOTA) is returned. In either case, the request is stopped. Normally, there are never more than three outstanding ASTs per LPA11-K request.

2.4.6 LPA11-K Data Transfer Stop Command

The Cancel I/O on Channel ($CANCEL) system service is used to abort data transfers for a particular process. When the LPA11-K driver receives a $CANCEL request, a data transfer stop command is issued to the LPA11-K.

To stop a data transfer, set bit 14 of the user status word. If this bit is set, the transfer stops at the end of the next buffer transaction (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*).

2.5 High-Level Language Interface

The operating system supports several program-callable procedures that provide access to the LPA11-K. The formats of these calls are documented in this manual for FORTRAN users. MACRO users must set up a standard argument block and issue the standard CALL procedure. (MACRO users can also access the LPA11-K directly through the use of the device-specific QIO functions described in Section 2.4.) Users of other high-level languages must specify the proper subroutine or procedure invocation.

2.5.1 High-Level Language Support Routines

There are 20 high-level language procedures for the LPA11-K. These procedures are divided into four classes. Table 2–4 lists the classes and the VAX procedures for the LPA11-K.

Table 2–4 Procedures for the LPA11-K

Class	Subroutine	Function
Sweep Control	LPA$ADSWP	Start A/D converter sweep
	LPA$DASWP	Start D/A converter sweep
	LPA$DISWP	Start digital input sweep
	LPA$DOSWP	Start digital output sweep
	LPA$LAMSKS	Specify LPA11-K controller and digital mask words
	LPA$SETADC	Specify channel select parameters
	LPA$SETIBF	Specify buffer parameters
	LPA$STPSWP	Stop sweep
Clock control	LPA$CLOCKA	Set Clock A rate
	LPA$CLOCKB	Set Clock B rate
	LPA$XRATE	Compute clock rate and preset value
Data Buffer	LPA$IBFSTS	Return buffer status
Control	LPA$IGTBUF	Return next available buffer
	LPA$INXTBF	Alter buffer order
	LPA$IWTBUF	Return next buffer or wait
	LPA$RLSBUF	Release buffer to LPA11-K
	LPA$RMVBUF	Remove buffer from device queue
Miscellaneous	LPA$CVADF	Convert A/D input to floating point
	LPA$FLT16	Convert unsigned integer to floating point
	LPA$LOADMC	Load microcode and initialize LPA11-K

2.5.1.1 Buffer Queue Control

This section is provided for informational purposes only.

Buffer queue control for data transfers by LPA11-K subroutines involves the use of the following queues:

- Device queue (DVQ)
- User queue (USQ)
- In-use queue (IUQ)

Each data transfer request can specify from one through eight data buffer areas. The user specifies these buffers by address. During execution of the request, the LPA11-K assigns an index from 0 through 7 when a buffer is referenced.

The DVQ contains the indexes of all the buffers that the user has released (buffers made available to be filled or emptied by the LPA11-K). For output functions (D/A and digital output), these buffers contain data to be output by the LPA11-K. For input functions (A/D and digital input), these buffers are empty and waiting to be filled by the LPA11-K.

The USQ contains the indexes of all buffers that are waiting to be returned to the user. The LPA$IWTBUF and LPA$IGTBUF calls are used to return the index of the next buffer in the USQ. For output functions (D/A and digital output), these buffers are empty and waiting to be filled by the application program. For input functions (A/D and digital input), these buffers contain data to be processed by the application program.

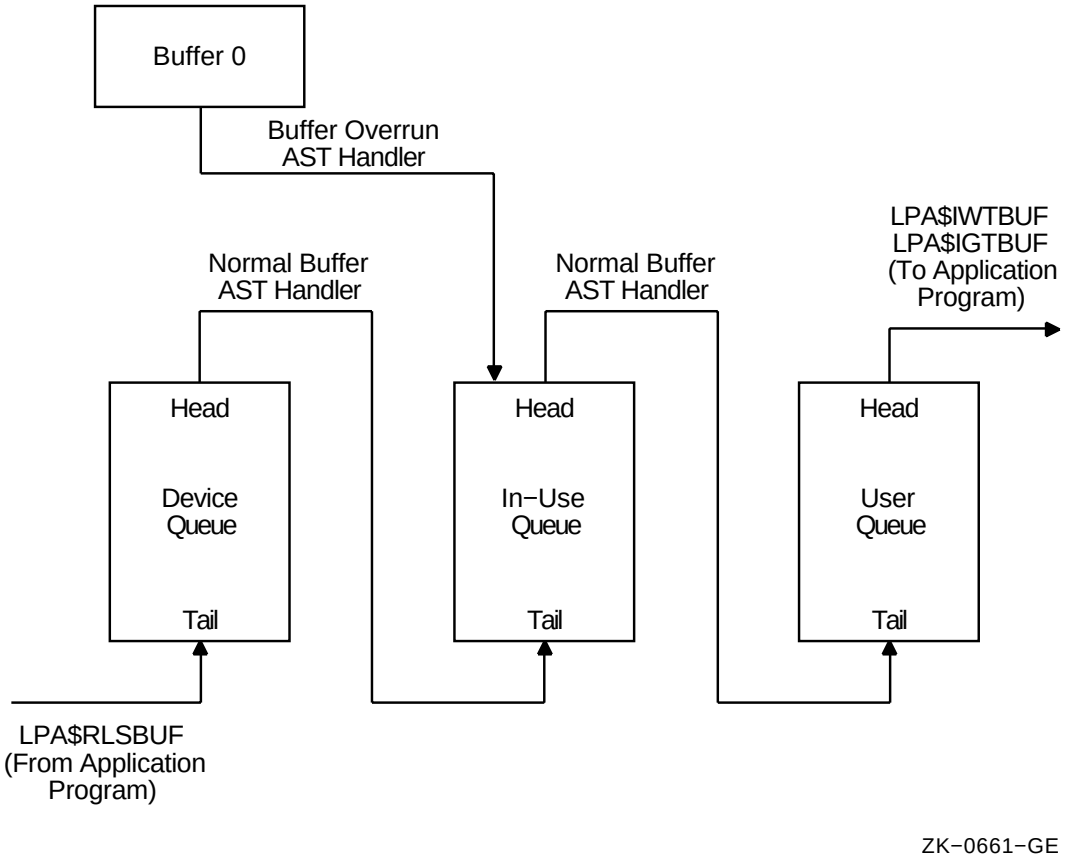
The IUQ contains the indexes of all buffers that are currently being processed by the LPA11-K. Normally, the IUQ contains the indexes of the following buffers:

- The buffer currently being filled or emptied by the LPA11-K
- The next buffer to be filled or emptied by the LPA11-K (this is the buffer specified by the next buffer index field in the user status word).

Because the LPA11-K driver requires that at least one buffer be ready when the input or output sweep is started, the user must call the LPA$RLSBUF subroutine before the sweep is initiated.

Figure 2–3 shows the flow between the buffer queues.

Figure 2–3 Buffer Queue Control



2.5.1.2 Subroutine Argument Usage

Table 2–5 describes the general use of the subroutine arguments. The subroutine descriptions in the following sections contain additional information on argument usage. The (IBUF), (BUF), and (ICHN) (random channel list address) arguments must be aligned on specific boundaries.

Table 2–5 Subroutine Argument Usage

Argument	Meaning
IBUF	A 50-longword array initialized by the LPA$SETIBF subroutine. IBUF is the impure area used by the buffer management subroutines. A unique IBUF array is required for each simultaneously active request. IBUF must be longword aligned. The first quadword in the IBUF array is an I/O status block (IOSB) for high-level language subroutines. The LPA$IGTBUF and LPA$IWTBUF subroutines fill this quadword with the current and completion status (see Section 2.6).
LBUF	Specifies the size of each data buffer in words (must be even for dedicated mode sweeps). All buffers are the same size. The minimum value for LBUF is 6 for multirequest mode data transfers and 258 for dedicated mode data transfers. The aggregate size of the assigned buffers must be less than 32,768 words. Thus, the maximum size of each buffer (in words) is limited to 32,768 divided by the number of buffers. The LBUF argument length is one word.
NBUF	Specifies the number of times the buffers are to be filled during the life of the request. If 0 (default) is specified, sampling is indefinite and must be stopped with the LPA$STPSWP subroutine. The NBUF argument length is one longword.
MODE	Specifies sampling options. MODE bit values are listed in the appropriate subroutine descriptions. The default is 0. MODE values can be added to specify several options. No options are mutually exclusive, although not all bits can be applicable at the same time. The MODE argument length is one word.
IRATE	Specifies the clock rate as follows:

Value	Meaning
–1	Direct-coupled Schmidt trigger 1 (Clock A only)
0	Clock B overflow or no rate
1	1 MHz
2	100 kHz
3	10 kHz
4	1 kHz
5	100 Hz
6	Schmidt trigger
7	Line frequency

	The IRATE argument length is one longword.
IPRSET	Specifies the hardware clock preset value. This value is the two's complement of the desired number of clock ticks between clock interrupts. (The maximum value is 0, the two's complement of 65,536.) IPRSET can be computed by the LPA$XRATE subroutine. The IPRSET argument length is one word.

(continued on next page)

Table 2–5 (Cont.) Subroutine Argument Usage

Argument	Meaning
DWELL	Specifies the number of hardware clock overflows between sample sequences in multirequest mode. For example, if DWELL is 20 and NCHN is 3, then after 20 clock overflows one channel is sampled on each of the next three successive overflows; no sampling occurs for the next 20 clock overflows. This allows different users to use different sample rates with the same hardware clock overflow rate. In dedicated mode, the hardware clock overflow rate controls sampling and DWELL is not accessed. Default for DWELL is 1. The DWELL argument length is one word.

IEFN	Specifies the event flag number or completion routine address. The selected event flag is set at the end of each buffer transaction. If IEFN is 0 (default), event flag 22 is used.
	IEFN can also specify the address of a completion routine. This routine is called by the buffer management routine when a buffer is available and when the request is terminated, either successfully or with an error. The standard calling and return sequences are used. The completion routine is called from an AST routine and is therefore at AST level.
	If IEFN specifies the address of a completion routine, the program must call the LPA$IGTBUF subroutine to obtain the next buffer. If IEFN specifies an event flag, the program must call the LPA$IWTBUF subroutine to obtain the next buffer and must use the %VAL operator:
	`,%VAL(3),  (Event flag 3)`
	`,BFRFULL,  (Address of completion routine)`
	The IEFN argument length is one longword.
	If multiple sweeps are initiated, they must use different event flags. The software does not enforce this policy.
	Event flag 23 is reserved for use by the LPA$CLOCKA and LPA$CLOCKB subroutines. If either of these subroutines is included in the user program, event flag 23 cannot be used. Also, if IEFN is defaulted, event flag 22 cannot be used in the user program.

LDELAY	Specifies the delay, in IRATE units, from the start event until the first sample is taken. The maximum value is 65,535; default is 1. The LDELAY argument length is one word. The LPA11-K supports the LDELAY argument in multirequest mode only.
ICHN	Specifies the number of the first I/O channel to be sampled. Default is channel 0. The ICHN argument length is one byte. The channel number is not the same as the channel assigned to the device by the $ASSIGN system service. The LPA11-K uses the channel number to specify the multiplexer address of an A/D, D/A, or digital I/O device on the LPA11-K internal I/O bus.
NCHN	Specifies the number of I/O device channels to sample in a sample sequence. Default is 1. If the NCHN argument is 1, the single channel bit is set in the mode word of the start request descriptor array (RDA) when the sweep is started. The RDA contains the information needed by the LPA11-K for each command (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*). The NCHN argument length is one word.
IND	Receives the success or failure code of the call. The IND argument length is one longword.

2.5.2 LPA$ADSWP — Initiate Synchronous A/D Sampling Sweep

The LPA$ADSWP subroutine initiates A/D sampling through an AD11-K.

The format of the LPA$ADSWP subroutine call is as follows:

```
CALL LPA$ADSWP (IBUF,LBUF,[NBUF],[MODE],[DWELL],[IEFN],-
                [LDELAY],[ICHN],[NCHN],[IND])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

MODE Specifies sampling options. The operating system defines the following sampling option values:

Value	Meaning
32	Parallel A/D conversion sample algorithm is used if dual A/D converters are specified (value = 8192). Absence of this bit implies the serial A/D conversion sample algorithm.
64	Multirequest mode request. Absence of this bit implies a dedicated mode request.
512	External trigger (Schmidt trigger 1). Dedicated mode only. This value is used when a user-supplied external sweep trigger is desired. The external trigger is supplied by the KW11-K (Schmidt trigger 1 output) to the AD11-K (external start input). If MODE=512, the user process must specify a Clock A rate of –1 for proper A/D sampling. This is nonclock-driven sampling (see Section 2.5.10). (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* provides additional information on the use of external triggers.)
1024	Time stamped sampling with Clock B. The double word consists of one data word followed by the value of the LPA11-K internal 16-bit counter at the time of the sample (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*). Multirequest mode only.
2048	Event marking. Multirequest mode only. (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes event marking.)
4096	Start method. If selected, the digital input start method is used. If not selected, the immediate start method is used. Multirequest mode only.
8192	Dual A/D converters are to be used. Dedicated mode only.
16384	Buffer overrun is a nonfatal error. The LPA11-K will automatically default to fill buffer 0 if a buffer overrun condition occurs.

If MODE is defaulted, A/D sampling starts immediately with absolute channel addressing in dedicated mode. The LPA11-K does not support delays in dedicated mode.

IND Returns the success or failure status as follows:

0 = Error in call. Possible causes are the following: LPA$SETIBF subroutine was not previously called; LPA$RLSBUF subroutine was not previously called; size of data buffers disagrees with the size computed by the LPA$SETIBF subroutine call.

1 = successful sweep started

nnn = status code

2.5.3 LPA$DASWP — Initiate Synchronous D/A Sweep

The LPA$DASWP subroutine initiates D/A output to an AA11-K.

The format for the LPA$DASWP subroutine call is as follows:

```
CALL LPA$DASWP (IBUF,LBUF,[NBUF],[MODE],[DWELL],[IEFN],-
                [LDELAY],[ICHN],[NCHN],[IND])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

MODE Specifies the sampling options. The operating system defines the following start criteria values:

Value	Meaning
0	Immediate start. This is the default value for MODE.
64	Multirequest mode. If not selected, this request is for dedicated mode.
4096	Start method. If selected, the digital input start method is used. If not selected, the immediate start method is used. Multirequest mode only.
16384	Buffer overrun is a nonfatal error. The LPA11-K will automatically default to empty buffer 0 if a buffer overrun condition occurs.

IND Returns the success or failure status as follows:

0 = Error in call. Possible causes are the following: LPA$SETIBF subroutine was not previously called; LPA$RLSBUF subroutine was not previously called; size of data buffers disagrees with the size computed by the LPA$SETIBF subroutine call.

1 = successful sweep started

nnn = status code

2.5.4 LPA$DISWP — Initiate Synchronous Digital Input Sweep

The LPA$DISWP subroutine initiates digital input through a DR11-K. It is applicable in multirequest mode only.

The format of the LPA$DISWP subroutine call is as follows:

```
CALL LPA$DISWP (IBUF,LBUF,[NBUF],[MODE],[DWELL],[IEFN],-
      [LDELAY],[ICHN],[NCHN],[IND])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

MODE Specifies sampling options. The operating system defines the following sampling option values:

Value	Meaning
0	Immediate start. This is the default value for MODE.
512	External trigger for DR11-K. (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes the use of external triggers.)
1024	Time stamped sampling with Clock B. The double word consists of one data word followed by the value of the internal LPA11-K 16-bit counter at the time of the sample (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*).
2048	Event marking. (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes event marking.)
4096	Start method. If selected, the start method is digital input. If not selected, the start method is immediate. Multirequest mode only.
16384	Buffer overrun is a nonfatal error. The LPA11-K will automatically default to fill buffer 0 if a buffer overrun condition occurs.

IND Returns the success or failure status as follows:

0 = Error in call. Possible causes are the following: LPA$SETIBF subroutine was not previously called; LPA$RLSBUF subroutine was not previously called; size of data buffers disagrees with the size computed by the LPA$SETIBF subroutine call.

1 = successful sweep started

nnn = status code

2.5.5 LPA$DOSWP — Initiate Synchronous Digital Output Sweep

The LPA$DOSWP subroutine initiates digital output through a DR11-K. It is applicable in multirequest mode only.

The format of the LPA$DOSWP subroutine call is as follows:

```
CALL LPA$DOSWP (IBUF,LBUF,[NBUF],[MODE],[DWELL],[IEFN],-
                [LDELAY],[ICHN],[NCHN],[IND])
```

Arguments are as described in Section 2.5.1.2, plus the following:

MODE Specifies sampling options. The operating system defines the following values:

Value	Meaning
0	Immediate start. This is the default value for MODE.
512	External trigger for DR11-K. (The *LPA11-K Laboratory Peripheral Accelerator User's Guide* describes the use of external triggers.)
4096	Start method. If selected, digital input start. If not selected, immediate start.
16384	Buffer overrun is a nonfatal error. The LPA11-K will automatically default to empty buffer 0 if a buffer overrun condition occurs.

IND Returns the success or failure status as follows:

0 = Error in call. Possible causes are the following: LPA$SETIBF subroutine was not previously called; LPA$RLSBUF subroutine was not previously called; size of data buffers disagrees with the size computed by the LPA$SETIBF subroutine call.

1 = Successful sweep started

nnn = Status code

2.5.6 LPA$LAMSKS — Set LPA11-K Masks and NUM Buffer

The LPA$LAMSKS subroutine initializes a user buffer that contains a number to append to the logical name LPA11$, a digital start word mask, an event mark mask, and channel numbers for the two masks.

The LPA$LAMSKS subroutine must be called in the following cases:

- If users intend to use digital input starting or event marking
- If users do not want to use the default of LAA0 assigned to LPA11$0
- If multiple LPA11-Ks are used

The format of the LPA$LAMSKS subroutine call is as follows:

```
CALL LPA$LAMSKS (LAMSKB,[NUM],[IUNIT],[IDSC],[IEMC],[IDSW],[IEMW],[IND])
```

Argument descriptions are as follows:

LAMSKB	Specifies a four-word array.
NUM	Specifies the number appended to LPA11$. The sweep is started on the LPA11-K assigned to LPA11$num.
IUNIT	Not used. This argument is present for compatibility only.
IDSC	Specifies the digital START word channel. Range is 0 through 4. The IDSC argument length is one byte.
IEMC	Specifies the event MARK word channel. Range is 0 through 4. The IEMC argument length is one byte.
IDSW	Specifies the digital START word mask. The IDSW argument length is one word.
IEMW	Specifies the event MARK word mask. The IEMW argument length is one word.
IND	Always equal to 1 (success). This argument is present for compatibility only.

2.5.7 LPA$SETADC — Set Channel Information for Sweeps

The LPA$SETADC subroutine establishes channel start and increment information for the sweep control subroutines (see Table 2–4). It must be called to initialize IBUF before the LPA$SETADC subroutine is called.

The LPA$SETADC subroutine can be called in either of the following formats:

```
CALL LPA$SETADC (IBUF,[IFLAG],[ICHN],[NCHN],[INC],[IND])
```

or

```
IND=LPA$SETADC (IBUF,[IFLAG],[ICHN],[NCHN],[INC])
```

Argument descriptions are as follows:

IND	Returns the success or failure status as follows: 0 = LPA$SETIBF was not called prior to the LPA$SETADC call 1 = LPA$SETADC call successful
IBUF	The IBUF array specified in the LPA$SETIBF call.
IFLAG	Reserved. This argument is present for compatibility only.
ICHN	Specifies the first channel number. Range is 0 through 255; default is 0. The ICHN argument length is one longword. If INC = 0, ICHN is the address of a random channel list. This address must be word aligned.
NCHN	Specifies the number of samples taken per sample sequence. Default is 1.
INC	Specifies the channel increment. Default is 1. If INC is 0, ICHN is the address of a random channel list. The INC argument length is one longword.

2.5.8 LPA$SETIBF — Set IBUF Array for Sweeps

The LPA$SETIBF subroutine initializes the IBUF array for use with the following subroutines:

LPA$ADSWP	LPA$DASWP	LPA$DISWP
LPA$DOSWP	LPA$IBFSTS	LPA$IGTBUF
LPA$INXTBF	LPA$IWTBUF	LPA$RLSBUF

LPA$RMVBUF LPA$SETADC LPA$STPSWP

The format of the LPA$SETIBF subroutine call is as follows:

```
CALL LPA$SETIBF (IBUF,[IND],[LAMSKB],BUF0,[BUF1,...,BUF7])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IBUF — Specifies a 50-longword array that is initialized by this subroutine. IBUF must be longword-aligned. (See Table 2–5 for additional information on IBUF.)

IND — Returns the success or failure status as follows:

0 = Error in call. Possible causes are the following: incorrect number of arguments; IBUF array not longword-aligned; buffer addresses not equidistant.

1 = IBUF initialized successfully.

LAMSKB — Specifies the name of a four-word array. This array allows the use of multiple LPA11-Ks within the same program because the argument used to start the sweep is specified by the LPA$LAMSKS subroutine call. (See Section 2.5.6 for a description of the LPA$LAMSKS subroutine.)

BUF0, . . . — Specify the names of the buffers. A maximum of eight buffers can be specified. At least two buffers must be specified to provide continuous sampling. The LPA11-K driver requires that all buffers be contiguous. To ensure this, the LPA$SETIBF subroutine verifies that all buffer addresses are equidistant. Buffers must be longword-aligned.

2.5.9 LPA$STPSWP — Stop In-Progress Sweep

The LPA$STPSWP subroutine allows you to stop a sweep that is in progress.

The format of the LPA$STPSWP subroutine call is as follows:

```
CALL LPA$STPSWP (IBUF,[IWHEN],[IND])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IBUF — The IBUF array specified in the LPA$ADSWP, LPA$DASWP, LPA$DISWP, or LPA$DOSWP subroutine call that initiated the sweep.

IWHEN — Specifies when to stop the sweep. The operating system defines the following values:

0 = Abort sweep immediately. Uses the $CANCEL system service. This is the default sweep stop.

1 = Stop sweep when the current buffer transaction is completed. (This is the preferred way to stop requests.)

IND — Receives a success or failure code in the following standard format:

1 = Success

nnn = Error code issued by the $CANCEL system service

Note that when the LPA$STPSWP subroutine is returned, the sweep cannot be stopped. If it is necessary to wait until the sweep has stopped, you can call the LPA$IWTBUF subroutine in a loop until it returns IBUFNO = –1 (see Section 2.5.16).

2.5.10 LPA$CLOCKA — Clock A Control

The LPA$CLOCKA subroutine sets the clock rate for Clock A.

The format of the LPA$CLOCKA subroutine call is as follows:

```
CALL LPA$CLOCKA (IRATE,IPRSET,[IND],[NUM])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IRATE — Specifies the clock rate. One of the following values must be specified:

Value	Meaning
–1	Direct-coupled Schmidt trigger 1. Used only for A/D sweeps in dedicated mode, that is, MODE = 512 (see Section 2.5.2).
0	Clock B overflow or no rate
1	1 MHz
2	100 kHz
3	10 kHz
4	1 kHz
5	100 Hz
6	Schmidt trigger 1
7	Line frequency

IPRSET — Specifies the clock preset value. Maximum of 16 bits. The LPA$XRATE subroutine can be used to calculate this value. The clock rate divided by the clock preset value yields the clock overflow rate.

IND — Receives a success or failure code as follows:

1 = Clock A set successfully

nnn = Error code indicating an I/O error

NUM — Specifies the number to be appended to the logical name LPA11$. The default value is 0. This subroutine sets Clock A on the LPA11-K assigned to LPA11$num.

2.5.11 LPA$CLOCKB — Clock B Control

The LPA$CLOCKB subroutine provides the user with control of the KW11-K Clock B.

The format of the LPA$CLOCKB subroutine call is as follows:

```
CALL LPA$CLOCKB ([IRATE],IPRSET,MODE,[IND],[NUM])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IRATE — Specifies the clock rate. One of the following must be specified:

Value	Meaning
0	Stops Clock B
1	1 MHz
2	100 kHz
3	10 kHz
4	1 kHz
5	100 Hz
6	Schmidt trigger 3
7	Line frequency

If IRATE is 0 (default), the clock is stopped and the IPRSET and MODE arguments are ignored.

IPRSET — Specifies the preset value by which the clock rate is divided to yield the overflow rate. Maximum of eight bits. Overflow events can be used to drive Clock A. The LPA$XRATE subroutine can be used to calculate the IPRSET value.

MODE — Specifies options. The operating system defines the following:

1 = Clock B operates in noninterrupt mode.

2 = The feed B to A bit in the Clock B status register will be set (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*).

IND — Receives a success or failure code as follows:

1 = Clock B set successfully

nnn = Error code indicating an I/O error

NUM — Specifies the number to be appended to the logical name LPA11$. The default value is 0. This subroutine sets Clock B on the LPA11-K assigned to LPA11$num.

2.5.12 LPA$XRATE — Compute Clock Rate and Preset Value

The LPA$XRATE subroutine computes the clock rate and preset value for the LPA$CLOCKA and LPA$CLOCKB subroutines using the specified intersample interval (AINTRVL).

The LPA$XRATE subroutine can be called in either of the following formats:

```
CALL LPA$XRATE (AINTRVL,IRATE,IPRSET,IFLAG)

ACTUAL=LPA$XRATE(AINTRVL,IRATE,IPRSET,IFLAG)
```

Arguments are as described in Section 2.5.1.2, with the following additions:

AINTRVL — Specifies the intersample time selected by the user. The time is expressed in decimal seconds. Data type is floating point.

IRATE — Receives the computed clock rate as a value from 1 through 5.

IPRSET — Receives the computed clock preset value.

IFLAG — If the computation is for Clock A, IFLAG is 0; if for Clock B, IFLAG is not 0 (the maximum preset value is 255). The IFLAG argument length is one byte.

ACTUAL	Receives the actual intersample time if called as a function. Data type is floating point. If there are truncation and round-off errors, the resulting intersample time can be different from the specified intersample time. Note that when the LPA$XRATE subroutine is called from VAX FORTRAN IV-PLUS programs as a function, it must be explicitly declared a real function. Otherwise, the LPA$XRATE subroutine defaults to an integer function.

If AINTRVL is either too large or too small to be achieved, both IRATE and ACTUAL are returned to 0.

2.5.13 LPA$IBFSTS — Return Buffer Status

The LPA$IBFSTS subroutine returns information on the buffers used in a sweep.

The format of the LPA$IBFSTS subroutine call is as follows:

```
CALL LPA$IBFSTS (IBUF,ISTAT)
```

Argument descriptions are as follows:

IBUF	The IBUF array specified in the call that initiated the sweep.
ISTAT	Specifies a longword array with as many elements as there are buffers involved in the sweep (maximum of eight). LPA$IBFSTS fills each array element with the status of the corresponding buffer: +2 = Buffer in device queue. LPA$RLSBUF has been called for this buffer. +1 = Buffer in user queue. The LPA11-K has filled (data input) or emptied (data output) this buffer. 0 = Buffer is not in any queue. –1 = Buffer is in the in-use queue; that is, it is either being filled or emptied, or it is the next to be filled or emptied by the LPA11-K.

2.5.14 LPA$IGTBUF — Return Buffer Number

The LPA$IGTBUF subroutine returns the number of the next buffer to be processed by the application program, the buffer at the head of the user queue (see Figure 2–3). It should be called by a completion routine at AST level to determine the next buffer to process. If an event flag was specified in the start sweep call, the LPA$IWTBUF, not the LPA$IGTBUF subroutine, should be called.

The LPA$IGTBUF subroutine can be called in one of these formats:

```
CALL LPA$IGTBUF (IBUF,IBUFNO)

IBUFNO=LPA$IGTBUF(IBUF)
```

Arguments are as described in Section 2.5.1.2, plus the following:

IBUF	The IBUF array specified in the call that initiated the sweep.
IBUFNO	Returns the number of the next buffer to be filled or emptied by the application program.

Table 2–6 lists the possible combinations of IBUFNO and IOSB contents on the return from a call to the LPA$IGTBUF subroutine. The first four words of the IBUF array contain the I/O status block (IOSB). If IBUFNO is –1, the IOSB must be checked to determine the reason.

Table 2–6 LPA$IGTBUF Call — IBUFNO and IOSB Contents

IBUFNO	IOSB(1)	IOSB(2)	IOSB(3),(4)	Meaning
n	0	(byte count)	0	Normal buffer complete.
–1	0	0	0	No buffers in queue. Request still active.
–1	1	0	0	No buffers in queue. Sweep terminated normally.
–1	Error code	0	LPA11-K ready-out and maintenance registers (only if SS$DEVREQERR, SS$_CTRLERR, or SS$DEVCMDERR is returned)	No buffers in queue. Sweep terminated due to error condition. Section 2.6 describes the error codes; the *LPA11-K Laboratory Peripheral Accelerator User's Guide* lists the LPA11-K error codes.

2.5.15 LPA$INXTBF — Set Next Buffer to Use

The LPA$INXTBF subroutine alters the normal buffer selection algorithm so that you can specify the next buffer to be filled or emptied. The specified buffer is reinserted at the head of the device queue.

The LPA$INXTBF subroutine can be called in one of these formats:

```
CALL LPA$INXTBF (IBUF,IBUFNO,IND) IND=LPA$INXTBF(IBUF,IBUFNO)
```

Arguments are as described in Section 2.5.1.2, plus the following:

IBUF	The IBUF array specified in the call that initiated the sweep.
IBUFNO	Specifies the number of the next buffer to be filled or emptied. The buffer must already be in the device queue.
IND	Returns the result of the call as follows: 0 = Specified buffer not in the device queue 1 = Next buffer successfully set

2.5.16 LPA$IWTBUF — Return Next Buffer or Wait

The LPA$IWTBUF subroutine returns the next buffer to be processed by the application program, the buffer at the head of the user queue. If the user queue is empty, the LPA$IWTBUF subroutine waits until a buffer is available. If a completion routine was specified in the call that initiated the sweep, LPA$IGTBUF, not LPA$IWTBUF, should be called.

The LPA$IWTBUF subroutine can be called in either of the following formats:

```
CALL LPA$IWTBUF (IBUF,[IEFN],IBUFNO) IBUFNO=LPA$IWTBUF(IBUF,[IEFN])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IBUF	The IBUF array specified in the call that initiated the sweep.
IEFN	Not used. This argument provides compatibility with the operating system. (The event flag is the one specified in the start sweep call.)
IBUFNO	Returns the number of the next buffer to be filled or emptied by the application program.

Table 2–7 lists the possible combinations of IBUFNO and I/O status block contents on the return from a call to the LPA$IWTBUF subroutine. The first four words of the IBUF array contain the I/O status block. If IBUFNO is –1, the I/O status block must be checked to determine the reason.

Table 2–7 LPA$IWTBUF Call — IBUFNO and IOSB Contents

IBUFNO	IOSB(1)	IOSB(2)	IOSB(3),(4)	Meaning
n	0	(byte count)	0	Normal buffer complete.
–1	1	0	0	No buffers in queue. Sweep terminated normally.
–1	Error code	0	LPA11-K ready-out and maintenance registers (only if SS$_DEVREQERR SS$_CTRLERR, or SS$_ DEVCMDERR is returned)	No buffers in queue. Sweep terminated due to error condition. Section 2.6 describes the error codes; the *LPA11-K Laboratory Peripheral Accelerator User's Guide* lists the LPA11-K error codes.

2.5.17 LPA$RLSBUF — Release Data Buffer

The LPA$RLSBUF subroutine declares one or more buffers available to be filled or emptied by the LPA11-K. It inserts the buffer at the tail of the device queue (see Figure 2–3).

The format of the LPA$RLSBUF subroutine call is as follows:

```
CALL LPA$RLSBUF (IBUF,[IND],INDEX0,INDEX1,...,INDEXN)
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IBUF — The IBUF array specified in the call that initiated the sweep.

IND — Returns the success or failure status as follows:

0 = Buffer number was illegal, the number of arguments specified was incomplete, or a double buffer overrun occurred. A double buffer overrun can occur only if buffer overrun was specified as a nonfatal error, a buffer overrun occurs, and buffer 0 was not released (probably on the user queue after a previous buffer overrun).

1 = Buffer(s) released successfully.

INDEX0, . . . — Specify the indexes (0-7) of the buffers to be released. A maximum of eight indexes can be specified.

The LPA$RLSBUF subroutine must be called to release a buffer (or buffers) to the device queue before the sweep is initiated. (See Section 2.5.1.1 for a discussion of buffer management.) Note that the LPA$RLSBUF subroutine does not verify whether the specified buffers are already in a queue. If a buffer is released when it is already in a queue, the queue pointers are invalidated and unpredictable results can occur.

If buffer overrun is specified as a nonfatal error, buffer 0 should not be released before the sweep is initiated. However, if either the LPA$IGTBUF or LPA$IWTBUF subroutine returns buffer 0, it should be released. In this case, buffer 0 is set aside (not placed on a queue) until the buffer overrun occurs. If a buffer overrun occurs and buffer 0 was not released, the LPA$RLSBUF subroutine returns an error the next time buffer 0 is released.

2.5.18 LPA$RMVBUF — Remove Buffer from Device Queue

The LPA$RMVBUF subroutine removes a buffer from the device queue.

The format of the LPA$RMVBUF subroutine call is as follows:

```
CALL LPA$RMVBUF (IBUF,IBUFNO,[IND])
```

Arguments are as described in Section 2.5.1.2, with the following additions:

IBUF	The IBUF array specified in the call that initiated the sweep.
IBUFNO	Specifies the number of the buffer to remove from the device queue.
IND	Returns the success or failure status as follows: 0 = Buffer not found in the device queue 1 = Buffer successfully removed from the device queue

2.5.19 LPA$CVADF — Convert A/D Input to Floating-Point

The LPA$CVADF subroutine converts A/D input values to floating-point numbers. It is supported to provide compatibility with the OpenVMS VAX operating system.

The LPA$CVADF subroutine can be called in either of the following formats:

```
CALL LPA$CVADF (IVAL,VAL) VAL=LPA$CVADF(IVAL)
```

Argument descriptions are as follows:

IVAL	Contains the value (bits 11:0) read from the A/D input. Bits 15:12 are 0.
VAL	Receives the floating-point value.

2.5.20 LPA$FLT16 — Convert Unsigned 16-Bit Integer to Floating-Point

The LPA$FLP16 subroutine converts unsigned 16-bit integers to floating point. It is supported to provide compatibility with the OpenVMS VAX operating system.

The LPA$FLT16 subroutine can be called in either of the following formats:

```
CALL LPA$FLT16 (IVAL,VAL) VAL=LPA$FLT16(IVAL)
```

Argument descriptions are as follows:

IVAL	An unsigned 16-bit integer.
VAL	Receives the converted value.

2.5.21 LPA$LOADMC — Load Microcode and Initialize LPA11-K

The LPA$LOADMC subroutine provides a program interface to the LPA11-K microcode loader. It sends a load request through a mailbox to the loader process to load microcode and to initialize an LPA11-K. (Section 2.7.1 describes the microcode loader process.)

The format of the LPA$LOADMC subroutine call is as follows:

```
CALL LPA$LOADMC ([ITYPE][,NUM][,IND][,IERROR])
```

Argument descriptions are as follows:

ITYPE — The type of microcode to be loaded. The operating system defines the following values:

Value	Meaning
1	Multirequest mode; default value
2	Dedicated A/D mode
3	Dedicated D/A mode

NUM — The number to be appended to the logical name LPA11$. The default value is 0.

IND — Receives the completion status as follows:

1 = Microcode loaded successfully

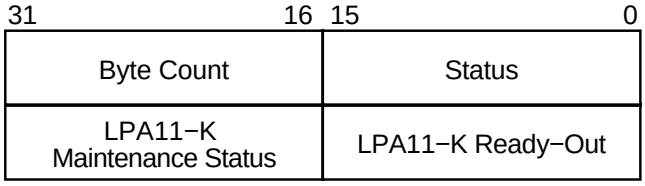
nnn = Error code

IERROR — Provides additional error information. Receives the second longword of the I/O status block if SS$_CTRLERR, SS$_DEVCMDERR, or SS$_DEVREQERR is returned in IND. Otherwise, the contents of IERROR are undefined.

2.6 I/O Status Block

The I/O status block (IOSB) format for the load microcode, start microprocessor, initialize LPA11-K, set clock, and start data transfer request QIO functions is shown in Figure 2–4.

Figure 2–4 I/O Functions IOSB Content



ZK−0662−GE

Status values and the byte count are returned in the first longword. Status values are defined by the $SSDEF macro. The byte count is the number of bytes transferred by a IO$_LOADMCODE request. If SS$_CTRLERR, SS$_DEVCMDERR, or SS$_DEVREQERR is returned in the status word, the second longword contains the LPA11-K ready-out register and LPA11-K maintenance status register values present at the completion of the request. The high byte of the ready-out register contains the specific LPA11-K error code (see the *LPA11-K Laboratory Peripheral Accelerator User's Guide*). Appendix A of this manual lists the status returns for LPA11-K I/O functions. (The OpenVMS system messages documentation provides explanations and suggested user actions for these returns.)

If high-level language library procedures are used, the status returns listed in Appendix A can be returned from the resultant QIO functions. Since buffers are filled by these procedures asynchronously, two I/O status blocks are provided in the IBUF array: one for the high-level language procedures and one for the

LPA11-K driver. The first four words of the IBUF array contain the I/O status block for the high-level language procedures.

2.7 Loading LPA11-K Microcode

The microcode loading and device initialization routines automatically load microcode during system initialization (if specified in the system manager's startup file) and power recovery. These routines also allow a nonprivileged user to load microcode and to restart the system.

The LPA11-K loader and initialization routines consist of the following parts:

- A microcode loader process that loads any of the three microcode versions, initializes the LPA11-K, and sets the clock rate. Loading is initiated by either a mailbox request or a power recovery AST. This process requires permanent mailbox (PRMMBX) and physical I/O privileges.
- An operator process that accepts operator commands or indirect file commands to load microcode and to initialize an LPA11-K. This process uses a mailbox to send a load request to the loader process; temporary mailbox (TMPMBX) privilege is required.
- An LPA11-K procedure library routine that provides a program interface to the LPA11-K microcode loader. The procedure sends a load request through a mailbox to the loader process to load microcode and to initialize an LPA11-K. Section 2.5.21 describes that routine in greater detail.

2.7.1 Microcode Loader Process

The microcode loader process loads microcode, initializes a specific LPA11-K, and sets the clock at the default rate (10 kHz interrupt rate). A bit set in a controller bit map indicates that the specified controller was loaded. The process specifies a power recovery AST, creates a mailbox whose name (LPA$LOADER) is entered in the system logical name table, and then hibernates.

The correct device configuration is determined automatically. When LPA11-K initialization is performed, every possible device (see Table 2–1) is specified as present on the LPA11-K. If the LPA11-K returns a "device not found" error, the LPA11-K is reinitialized with that device omitted.

On receipt of a power recovery AST, the loader process examines the controller bit map to determine which LPA11-Ks have been loaded. For each LPA11-K, the loader process performs the following functions:

- Obtains device characteristics
- Reloads the microcode previously loaded
- Reinitializes the LPA11-K
- Sets Clock A to the previous rate and preset value

2.7.2 Operator Process

The operator process loads microcode and initializes an LPA11-K through either terminal or indirect file commands. To run the operator process, type RUN SYS$SYSTEM:LALOAD. The command input syntax is as follows:

```
devname/type
```

In the preceding example, devname is the device name of the LPA11-K to be loaded. A logical name can be specified. However, only one level of logical name translation is performed. If devname is omitted, LAA0 is the default name. If /type appears, it specifies one of the following types of microcode to load:

- /MULTI_REQUEST—Multirequest mode
- /ANALOG_DIGITAL—Dedicated A/D mode
- /DIGITAL_ANALOG—Dedicated D/A mode

If /type type is omitted, /MULTI_REQUEST is the default.

After receiving the command, the operator process formats a message and sends it to the loader process. Completion status is returned through a return mailbox.

2.8 RSX–11M/M–PLUS and OpenVMS VAX Differences

This section lists those areas of the OpenVMS VAX high-level language support routines that differ from the RSX–11M LPA11-K routines. The *RSX–11M /M–PLUS I/O Drivers Reference Manual* provides a detailed description of the RSX–11M LPA11-K support routines. Differences between the OpenVMS VAX and RSX–11M/M–PLUS routines can be determined by comparing the descriptions in the *RSX–11M/M–PLUS I/O Drivers Reference Manual* with the descriptions for the routines in the preceding sections of this chapter.

2.8.1 General

The following are general features of OpenVMS VAX high-level support routines:

- The LUN argument is not used. The NUM argument specifies the number to be appended to the logical name LPA11$.
- All routine names have the prefix LPA$.
- In the LPA$SETIBF routine, buffer addresses are checked for contiguity.
- In the LPA$LAMSKS routine, the IUNIT argument is not used.
- In the LPA$IWTBUF routine, the IEFN argument is not used. The event flag specified in the sweep routine is used.
- The combinations of IBUFNO and I/O status block (IOSB) values returned by the LPA$IWTBUF and LPA$IGTBUF subroutines are different.

2.8.2 Alignment and Length

The following are features of alignment and length in OpenVMS VAX high-level support routines:

- Buffers must be contiguous.
- Buffers must be longword-aligned.
- The random channel list (RCL) must be word-aligned.
- The IBUF array length is 50 longwords and must be longword-aligned.

2.8.3 Status Returns

The following are features of status returns in OpenVMS VAX high-level support routines:

- The I/O status block (IOSB) length is eight bytes; numeric values of errors differ.
- Several routines return the following:

  1 = Success

  0 = Failure detected in support routine

  nnn = Status code; failure detected in system service

2.8.4 Sweep Routines

The following are features of sweep routines in OpenVMS VAX high-level support routines:

- If an event flag is specified, it must be within a %VAL( ) construction.
- A tenth argument, IND, is added to return the success or failure status.

2.9 LPA11-K Programming Examples

The following programming examples use LPA11-K high-level language procedures and LPA11-K Queue I/O functions.

The *VMS Device Support Manual* volume contains information that is applicable to LPA11-K programming.

2.9.1 LPA11-K High-Level Language Program (Program A)

This sample program (Example 2–1) is an example of how the LPA11-K high-level language procedures perform an A/D sweep using three buffers. The program uses default arguments whenever possible to illustrate the simplest possible calls. The program assumes that dedicated mode microcode has previously been loaded into the LPA11-K. Table 2–8 lists the variables used in this program.

Table 2–8 Program A Variables

Variable	Description
BUFFER	The data buffer array. BUFFER is a common area to guarantee longword alignment.
IBUF	The LPA11-K high-level language procedures use the IBUF array for local storage.
BUFNUM	BUFNUM contains the buffer number returned by LPA$IWTBUF. In this example, the possible values are 0, 1, and 2.
ISTAT	ISTAT contains the status return from the high-level language calls.

Example 2–1 LPA11-K High-Level Language Program (Program A)

```
C  ***************************************************************
C
C                              PROGRAM A
C
C  ***************************************************************

        INTEGER*2      BUFFER(1000,0:2),IOSB(4)
        INTEGER*4      IBUF(50),ISTAT,BUFNUM

        COMMON/AREA1/BUFFER

        EQUIVALENCE    (IOSB(1),IBUF(1))

C
C  Set clock rate to 1 khz, clock preset to -10.
C
        CALL LPA$CLOCKA(4,-10,ISTAT)
        IF (.NOT. ISTAT) GO TO 950
C
C  Initialize IBUF array for sweep.
C
        CALL LPA$SETIBF(IBUF,ISTAT,,BUFFER(1,0),BUFFER(1,1),BUFFER(1,2))
        IF (.NOT. ISTAT) GO TO 950
C
C  Release all the buffers. Note use of buffer numbers rather than
C  buffer names.
C
        CALL LPA$RLSBUF(IBUF,ISTAT,0,1,2)
        IF (.NOT. ISTAT) GO TO 950
C
C  Start A/D sweep
C
        CALL LPA$ADSWP(IBUF,1000,50,,,,,,ISTAT)
        IF (.NOT. ISTAT) GO TO 950
C
C  Get next buffer filled with data. If BUFNUM is negative, there
C  are no more buffers and the sweep is stopped.
C
100     BUFNUM = LPA$IWTBUF(IBUF)
        IF (BUFNUM .LT. 0) GO TO 800
C
C  Process data in buffer (1,BUFNUM) to buffer (1000,BUFNUM).
        .
        .
        .
(Application-dependent code is inserted at this point.)
        .
        .
        .
C  Release buffer is filled again.
C
200     CALL LPA$RLSBUF(IBUF,ISTAT,BUFNUM)
        IF (.NOT. ISTAT) GO TO 950
        GO TO 100
C
C  There are no more buffers to process. Check to ensure that the
C  sweep ended successfully. IOSB(1) contains either 1 or a
C  VMS status code.
C
800     IF (.NOT. IOSB(1)) CALL LIB$STOP(%VAL(IOSB(1)))
        PRINT *,'SUCCESSFUL COMPLETION'
```

(continued on next page)

Example 2–1 (Cont.) LPA11-K High-Level Language Program (Program A)

```
        GO TO 2000
C
C  Error return from subroutine. ISTAT contains either 0 or a
C  VMS error code.
C
950     IF (ISTAT .NE. 0) CALL LIB$STOP(%VAL(ISTAT))
        PRINT   *,'ERROR IN LPA11-K SUBROUTINE CALL'
2000    STOP
        END
C  ***************************************************************
```

2.9.2 LPA11-K High-Level Language Program (Program B)

This program (Example 2–2) is a more complex example of LPA11-K operations performed by the LPA11-K high-level language procedures. The following operations are demonstrated:

- Program-requested loading of LPA11-K microcode
- Setting the clock at a specified rate
- Use of nondefault arguments whenever possible
- An A/D sweep that uses an event flag
- A D/A sweep that uses a completion routine
- Buffer overrun set (buffer overrun is a nonfatal error)
- Random channel list (RCL) addressing
- Sequential channel addressing

Table 2–9 lists the variables used in this program.

Table 2–9 Program B Variables

Variable	Description
AD	An array of buffers for an A/D sweep (8 buffers of 500 words each)
DA	An array of buffers for a D/A sweep (2 buffers of 2000 words each)
IBUFAD	The IBUF array for an A/D sweep
IBUFDA	The IBUF array for a D/A sweep
RCL	The array that contains the random channel list (RCL)
ADIOSB	The array that contains the I/O status block for the A/D sweep. Equivalenced to the beginning of IBUFAD
DAIOSB	The array that contains the I/O status block for the D/A sweep. Equivalenced to the beginning of IBUFDA
ISTAT	Contains the status return from the high-level language calls

Example 2–2 LPA11-K High-Level Language Program (Program B)

```
C  ****************************************************************
C
C                             Program B
C
C  ****************************************************************

       EXTERNAL FILLBF
       REAL*4 LPA$XRATE

       INTEGER*2 AD(500,0:7),DA(2000,0:1),RCL(5),MODE,IPRSET
       INTEGER*2 ADIOSB(4),DAIOSB(4)

       INTEGER*4 IBUFAD(50),IBUFDA(50),LAMSKB(2)
       INTEGER*4 ISTAT,IERROR,IRATE,BUFNUM

       REAL*4 PERIOD

       COMMON  /SWEEP/AD,DA,IBUFAD,IBUFDA

       EQUIVALENCE (IBUFAD(1),ADIOSB(1)),(IBUFDA(1),DAIOSB(1))

       PARAMETER MULTI=1, HBIT='8000'X, LSTCHN=HBIT+7
C
C  Set up random channel list. Note that the last word must have bit
C  15 set.
C
       DATA RCL/2,6,3,4,LSTCHN/
C  ****************************************************************
C
C  Load multirequest mode microcode and set the clock overflow rate
C  to 5 khz.
C
C  ****************************************************************
C
C  Load microcode on LPA11-K assigned to LPA11$3.
C
       CALL LPA$LOADMC(MULTI,3,ISTAT,IERROR)
       IF (.NOT. ISTAT) GO TO 5000
C
C  Compute clock rate and preset. Set clock 'A' on LPA11-K
C  assigned to LPA11$3.
C
       PERIOD = LPA$XRATE(.0002,IRATE,IPRSET,0)
       IF (PERIOD .EQ. 0.0) GO TO 5500

       CALL LPA$CLOCKA(IRATE,IPRSET,ISTAT,3)
       IF (.NOT. ISTAT) GO TO 5000
C  ****************************************************************
C
C  Set up for A/D sweep
C
C  ****************************************************************
C
C  Initialize IBUF array. Note the use of the LAMSKB argument because
C  the LPA11-K assigned to LPA11$3 is used.
C
       CALL LPA$SETIBF(IBUFAD,ISTAT,LAMSKB,AD(1,0),AD(1,1),AD(1,2),
       1               AD(1,3),AD(1,4),AD(1,5),AD(1,6),AD(1,7))
       IF (.NOT. ISTAT) GO TO 5000
```

(continued on next page)

Example 2–2 (Cont.) LPA11-K High-Level Language Program (Program B)

```
        CALL LPA$LAMSKS(LAMSKB,3)
C
C  Set up random channel list sampling (20 samples in a sample
C  sequence).
C
        CALL LPA$SETADC(IBUFAD,,RCL,20,0,ISTAT)
        IF (.NOT. ISTAT) GO TO 5000
C
C  Release buffers for A/D sweep. Note that buffer 0 is not
C  released because buffer overrun will be specified as nonfatal.
C
        CALL LPA$RLSBUF(IBUFAD,ISTAT,1,2,3,4,5,6,7)
        IF (.NOT. ISTAT) GO TO 5000
C  ****************************************************************
C
C  Set up for D/A sweep
C
C  ****************************************************************
C
C  Note that the same LAMSKB array can be used because the LAMSKB
C  contents apply to both A/D and D/A sweeps.
C
        CALL LPA$SETIBF(IBUFDA,ISTAT,LAMSKB,DA(1,0),DA(1,1))
        IF (.NOT. ISTAT) GO TO 5000
C
C  Set up sampling parameters as follows:  initial channel = 1.
C  Number of channels sampled each sample sequence = 2, channel
C  increment = 2, that is, sample channels 1 and 3 each sample
C  sequence.
C
        CALL LPA$SETADC(IBUFDA,,1,2,2,ISTAT)
        IF (.NOT. ISTAT) GO TO 5000
C
C  Fill buffers with data for output to D/A.
C
        .
        .
        .
(Application-dependent code is inserted here to fill buffers
DA(1,0) through DA(2000,0) and DA(1,1) through DA(2000,1) with data).
        .
        .
        .
C
C  Release buffers for D/A sweep.
C
        CALL LPA$RLSBUF (IBUFDA,ISTAT,0,1)
        IF (.NOT. ISTAT) GO TO 5000
```

(continued on next page)

Example 2–2 (Cont.) LPA11-K High-Level Language Program (Program B)

```
C  ****************************************************************
C
C  Start both sweeps
C
C  ****************************************************************
C
C  Start A/D sweep. Mode bits specify buffer overrun is nonfatal and
C  multirequest mode. Sweep arguments specify 500 samples/buffer,
C  Indefinite sampling, dwell = 10 clock overflows, synchronize using
C  event flag 15, and a delay of 50 clock overflows.
C
       MODE = 16384 + 64
       CALL LPA$ADSWP(IBUFAD,500,0,MODE,10,%VAL(15),50,,,ISTAT)
       IF (.NOT. ISTAT) GO TO 5000
C
C  Start D/A sweep.  Mode specifies multirequest mode.  Other
C  arguments specify 2000 samples/buffer, fill 15 buffers, dwell = 25
C  clock overflows, synchronize by calling the completion routine
C  'FILLBF', and delay = 10 clock overflows.  (See the FILLBF listing
C  after the program B listing.)
C
       MODE = 64
       CALL LPA$DASWP(IBUFDA,2000,15,MODE,25,FILLBF,10,,,ISTAT)
       (.NOT.  ISTAT) GO TO 5000
C  ****************************************************************
C
C  Wait for an A/D buffer and then process the data it contains.  D/A
C  buffers are filled asynchronously by the completion routine FILLBF.
C
C  ****************************************************************
C
C  Wait for a buffer to be filled by A/D.  If BUFNUM is less than
C  zero, the sweep has stopped (either successfully or with an error).
C
100    BUFNUM = LPA$IWTBUF(IBUFAD)
       IF (BUFNUM .LT.  0) GO TO 1000
C
C  There is A/D data in AD(1,BUFNUM) through AD(500,BUFNUM)
C

       .
       .
       .
(Process the A/D data with the application-dependent code inserted
here.)
       .
       .
       .
```

(continued on next page)

Example 2–2 (Cont.) LPA11-K High-Level Language Program (Program B)

```
C
C  Assume sweep should be stopped when the last sample in buffer
C  equals 0.  Note that the sweep actually stops when the buffer
C  currently being filled is full.  Also note that LPA$IWTBUF
C  continues to be called until there are no more buffers to process.
C
        IF (AD(500,BUFNUM) .NE.  0) GO TO 200
        CALL LPA$STPSWP(IBUFAD,1,ISTAT)
        IF (.NOT.  ISTAT) GO TO 5000

C
C  After the data is processed, the buffer is released to be
C  filled again.  Then the next buffer is obtained from A/D.
C
200     CALL LPA$RLSBUF(IBUFAD,ISTAT,BUFNUM)
        IF (.NOT.  ISTAT) GO TO 5000
        GO TO 100

C
C  Enter here when A/D sweep has ended.  Check for error or
C  successful end.  (Note: Assume that the D/A sweep has already
C  ended - see completion routine FILLBF.)
C
1000    IF(ADIOSB(1)) GO TO 6000
        CALL LIB$STOP(%VAL(ADIOSB(1)))

C
C  Enter here if there was an error returned from one of the
C  LPA11-K high-level language calls.  ISTAT contains either 0
C  or a VMS status code.
C
5000    IF (ISTAT .NE.  0) CALL LIB$STOP (%VAL(ISTAT))
5500    PRINT *,'ERROR IN LPA11-K SUBROUTINE CALL'
        GO TO 7000

6000    PRINT *,'SUCCESSFUL COMPLETION'
7000    STOP
        END
C  ***************************************************************
C
C  Subroutine FILLBF
C
C  ***************************************************************
C
C  The FILLBF subroutine is called whenever the D/A has emptied a
C  buffer, and that buffer is available to be refilled.  This
C  subroutine gets the buffer, fills it, and releases it back to the
C  LPA11-K.  Note that the D/A sweep is stopped automatically after
C  15 buffers have been filled.  Also note that FILLBF is called by
C  an AST handler.  It is therefore called asynchronously from the
C  main program at AST level.  Care should be exercised when accessing
C  variables that are common to both levels.
C

        INTEGER*2  AD(500,0:7),DA(2000,0:1),DAIOSB(4)
        INTEGER*4  IBUFAD(50),IBUFDA(50),BUFNUM,ISTAT
        EQUIVALENCE  (IBUFDA(1),DAIOSB(1))
        COMMON  /SWEEP/AD,DA,IBUFAD,IBUFDA
```

(continued on next page)

Example 2–2 (Cont.) LPA11-K High-Level Language Program (Program B)

```
C
C  Get buffer number of next buffer to fill.
C
        BUFNUM = LPA$IGTBUF(IBUFDA)
        IF (BUFNUM .LT.  0) GO TO 3000

C
C  Fill buffer with data for output to D/A.
        .
        .
        .
(Application-dependent code is inserted here to fill buffer
DA(1,BUFNUM) through DA(2000,BUFNUM) with data.)
        .
        .
        .
C
C  Release buffer
C
        CALL LPA$RLSBUF(IBUFDA,ISTAT,BUFNUM)
        GO TO 4000

C
C  Check for successful end of sweep.
C
3000    IF(DAIOSB(1)) GO TO 4000

C
C  Error in sweep
C
        CALL LIB$STOP(%VAL(DAIOSB(1)))

4000    RETURN
        END
C  ********************************************************************
```

2.9.3 LPA11-K QIO Functions Program (Program C)

This sample program (Example 2–3) uses QIO functions to start an A/D data transfer from an LPA11-K. (The program assumes multirequest mode microcode has been loaded.) Sequential channel addressing is used. The data transfer is stopped after 100 buffers have been filled; no action is taken with the data as the buffers are filled. Note that this program starts the data transfer and then waits until the QIO operation completes.

Example 2–3 LPA11-K QIO Functions Program (Program C)

```
; ********************************************************************
;
;                              Program C
;
; ********************************************************************

        .TITLE  LPA11-K EXAMPLE PROGRAM
        .IDENT  /V01/

        .PSECT  LADATA,LONG
```

(continued on next page)

Example 2–3 (Cont.) LPA11-K QIO Functions Program (Program C)

```
IOSB:   .BLKQ  1                        ; I/O status block
COUNT:  .LONG  0                        ; Count of buffers filled

CBUFF:                                  ; Command buffer for start
                                        ; Data QIO
        .WORD  ^X20A                    ; Mode = Sequential channel
                                        ; Addressing, A/D,
                                        ; multirequest mode
        .WORD  3                        ; Valid buffer mask
                                        ; (4 buffers)
        .LONG  USW                      ; User Status Word address
        .LONG  4000                     ; Aggregate buffer length
        .LONG  DATA_BUFFER0             ; Address of data buffers
        .LONG  0                        ; No random channel list
                                        ; length
        .LONG  0                        ; No random channel list
                                        ; address
        .WORD  10                       ; Delay
        .BYTE  0                        ; Start channel
        .BYTE  1                        ; Channel increment
        .WORD  16                       ; Number of samples in
                                        ; sample sequence
        .WORD  1                        ; Dwell
        .BYTE  0                        ; Start word number
        .BYTE  0                        ; Event mark word
        .WORD  0                        ; Start word mask
        .WORD  0                        ; Event mark mask
        .WORD  0                        ; Fills out command buffer

USW:    .WORD  0                        ; User Status Word

        .ALIGN LONG                     ; Buffers must be
                                        ; longword aligned
DATA_BUFFER0: .BLKW    500              ; Data buffers
DATA_BUFFER1: .BLKW    500
DATA_BUFFER2: .BLKW    500
DATA_BUFFER3: .BLKW    500
DEVNAME: .ASCID /LAA0/

CHANNEL: .BLKW  1                       ; Contains channel number

        .PSECT  LACODE,NOWRT

START:  .ENTRY  START,^m<>
        $ASSIGN_S DEVNAM=DEVNAME,CHAN=CHANNEL ; Assign channel
        BLBS    R0,5$                   ; No error
        BRW     ERROR                   ; Error

5$:                                     ; Set clock overflow rate
                                        ; to 2 khz.  (1 mhz rate
                                        ; divided by 500 preset)
        $QIOW_S ,CHAN=CHANNEL,FUNC=#IO$_SETCLOCK,-
                IOSB=IOSB,,,,P2=#1,P3=#^X143,P4#-500
        BLBC    R0,ERROR                ; Error
        MOVZWL  IOSB,R0                 ; Pick up I/O status
        BLBC    R0,ERROR                ; Error
                                        ; Start data transfer
        CLRW    USW                     ; Clear USW (start with
                                        ; buffer 0)
        MOVL    #100,COUNT              ; Fill 100 buffers
        $QIOW_S ,CHANNEL,#IO$_STARTDATA,-
                IOSB=IOSB,,,P1=CBUFF,P2=#40,P3=#BFRAST
        BLBC    R0,ERROR                ; Error
```

(continued on next page)

Example 2–3 (Cont.) LPA11-K QIO Functions Program (Program C)

```
; Note that the QIO waits until it finishes.  Normally, the data is
; processed here as the buffers are filled.  Check for error when
; the QIO completes.

        MOVZWL   IOSB,R0                ; Pick up I/O status
        BLBC     R0,ERROR               ; Error
        RET                             ; All done - exit

ERROR:                                  ; Enter here if error
                                        ; status in R0
        PUSHL    R0                     ; Push onto stack
        CALLS    #1,G^LIB$STOP          ; Signal error

BFRAST: BFRAST,m^<>                     ; Buffer AST routine
                                        ; BFRAST is called whenever
                                        ; a buffer is filled
        .WORD    0
        INCB     USW+1                  ; Add 1 to buffer number
        CMPZV    #0,#3,USW+1,#3         ; Handle wraparound

        BLEQ     10$
        CLRB     USW+1                  ; Use buffer 0
10$:    DECL     COUNT                  ; Decrement buffer count
        BGTR     20$
        BISB     #^X40,USW+1            ; Enough buffers filled -
                                        ; Set stop bit
20$:    BICB     #^X80,USW+1            ; Clear done bit
        RET

        .END     START

; ****************************************************************
```

3
Line Printer Driver

This chapter describes the use of the line printer drivers LPDRIVER and LCDRIVER.

3.1 Supported Line Printer Devices

The following sections describe the line printer controllers and line printers supported by the operating system.

3.1.1 LP11 Line Printer Controller

The LP11 line printer controller provides an interface between the UNIBUS adapter and the line printer. The LP11 performs the following functions:

- Synchronizes single-character data transfers from the UNIBUS to the printer
- Informs the system about printer status
- Enables the printer to gain control of the UNIBUS to report interrupts

3.1.2 DMF32 and DMB32 Line Printer Controllers

The DMF32 and DMB32 line printer controllers provide a direct memory access (DMA) interface between the UNIBUS adapter (for the DMF32), or the VAXBI adapter (for the DMB32), and the line printer. The DMF32/DMB32 optionally perform the following functions:

- Tab expansion
- Carriage control
- Line wrapping and truncation
- Case conversion
- Passall mode
- Printall mode

3.1.3 LP27 Line Printer

The LP27 line printer is a high-speed, 132-column line printer, available with either a 64- or 96-character ASCII print set. The LP27-U is a fully buffered model that operates at a standard speed of up to 1200 lines per minute. Forms with up to six parts can be used for multiple copies. A version of the LP27 is available for operation of the printer up to 24.5 meters (1000 feet) from the host.

3.1.4 LA11 DECprinter I

The LA11 DECprinter I is a medium-speed printer that operates at a standard speed of 180 characters per second. It provides a forms length switch to set the top of form to any of 11 common lengths, a paper-out switch and alarm, and a variable forms width. The LA11 uses a 96-character ASCII set; the column width is 132 characters.

3.1.5 LN01 Laser Page Printer

The LN01 laser page printer is a nonimpact printer that employs laser technology to produce high-quality print. Using electrophotographic imaging and xerographic printing, the LN01 prints one page at a time at a rate of 12 pages per minute. The print resolution of 300 x 300 dots per square inch produces characters of even density and alignment. The LN01 uses two 188-character, fixed-space fonts; the column width is 132 characters.

3.1.6 LN03 Laser Page Printer

The LN03 laser page printer is a table-top, nonimpact page printer that uses laser imaging and xerographic printing techniques. The LN03 has a printing speed of eight pages per minute with a print resolution of 300 x 300 dots per square inch. Four built-in fonts are available. Several column widths, including 80 or 132 characters, are also available.

3.2 Driver Features

The line printer drivers provide output character formatting and error recovery. These features are described in the following sections.

3.2.1 Output Character Formatting

In write virtual and write logical block operations, user-supplied characters are output as follows (write physical block data is not formatted, but output directly):

- Rubouts are discarded.
- Tabs move the horizontal print position to the next MODULO (8) position unless the LP$M_TAB characteristic is clear.
- All lowercase alphabetic characters are converted to uppercase before printing (unless the characteristic specifying lowercase characters is set; see Section 3.4.3 and Table 3–2).
- On printers where the line-feed, form-feed, vertical-tab, and carriage-return characters empty the printer buffer, returns are held back and output only if the next character is not a form feed, line feed, or vertical tab. Carriage returns are always output on units that have the LP$M_CR characteristic set (see Section 3.4.3 and Table 3–2).
- The horizontal print position is incremented on the output of all characters, including the space character. Characters are discarded if the horizontal print position is equal to or greater than the carriage width, unless the LP$M_WRAP characteristic is set or the LP$M_TRUNCATE characteristic is clear (see Section 3.3).
- On printers without a mechanical form feed (the form-feed function characteristic is not set; see Section 3.4.3 and Table 3–2), a form feed is converted to multiple line feeds. The number of line feeds is based on the current line count and the page length.

- Print lines are counted and returned to the caller in the second longword of the I/O status block.

3.2.2 Error Recovery

The VMS line printer drivers perform the following error recovery operations:

- If the printer is off line for 30 seconds, a "device not ready" message is sent to the system operator process.
- If the printer runs out of paper or has a fault condition, a "device not ready" message is sent to the system operator after 30 seconds. Successive messages, if they occur, are sent 1, 2, 4, 8, . . . minutes after the initial message.
- The current operation is retried every two seconds to test for a changed situation, such as the printer coming on line.
- The current I/O operation can be canceled at the next timeout without the printer being on line.
- When the printer comes on line, device operation resumes automatically.

3.3 Line Printer Driver Device Information

You can obtain information on printer characteristics by using the Get Device /Volume Information ($GETDVI) system service. (See the *OpenVMS System Services Reference Manual*.)

$GETDVI returns line printer characteristics when you specify the item codes DVI$_DEVCHAR and DVI$_DEVDEPEND. Tables 3–1 and 3–2 list these characteristics. The $DEVDEF macro defines the device-independent characteristics; the $LPDEF macro defines the device-dependent characteristics. DVI$_DEVDEPEND returns a longword field that contains the device-dependent characteristics in the three low-order bytes and the page length in the high-order byte. Maximum page length is 255.

DVI$_DEVTYPE and DVI$_DEVCLASS return the device type and class names, which are defined by the $DCDEF macro. The device type is a value that corresponds to the printer, for example, LP$_LP27 or LP$_LA11. The device class for printers is DC$_LP. DVI$_DEVBUFSIZ returns the page width, which is a value in the range of 0 through 255 on a DMF32 controller and 0 through 65535 on an LP11 or a DMB32 controller.

Table 3–1 Printer Device-Independent Characteristics

Characteristic[1]	Meaning
Dynamic Bits (Conditionally Set)	
DEV$M_SPL	Device is spooled.
DEV$M_AVL	Printer is on line and available.
Static Bits (Always Set)	
DEV$M_REC	Device is record-oriented.

[1]Defined by the $DEVDEF macro.

(continued on next page)

Table 3–1 (Cont.) Printer Device-Independent Characteristics

Characteristic[1]	Meaning
	Static Bits (Always Set)
DEV$M_CCL	Carriage control is enabled.
DEV$M_ODV	Device is capable of output.

[1]Defined by the $DEVDEF macro.

Table 3–2 Device-Dependent Characteristics for Line Printers

Value[1]	Meaning
LP$M_CR	Printer requires carriage return (see Section 3.2.1).
LP$M_FALLBACK	Printer translates multinational characters to a 7-bit equivalent representation if possible. Otherwise, an underscore character (_) replaces the character. LPM$M_FALLBACK has no effect on physical block operations. See the *OpenVMS I/O User's Reference Manual* for a list of multinational characters.
LP$M_LOWER	Printer can print lowercase characters. If this value is not set, all lowercase characters are converted to uppercase when output. (LP$M_LOWER has no effect on write physical block operations.)
LP$M_MECHFORM	Printer has mechanical form feed. This characteristic is used when variable form length is required, such as in check printing. Driver sends ASCII form feed (decimal 12). Otherwise, multiple line feeds are generated. The page length determines the number of line feeds.
LP$M_PASSALL	All output data is in binary (no data interpretation occurs). Data termination occurs when the buffer is full (default buffer size is 132 bytes). Character formatting is disabled.
LP$M_PRINTALL	All printing and nonprinting characters are transferred to the printer, while character formatting remains enabled.
LP$M_TAB	Printer enables tab expansion.
LP$M_TRUNCATE	Printer truncates records that are larger than the carriage width.
LP$M_WRAP	Printer wraps records that are larger than the carriage width. If a string of text is longer than the width specified in the second longword, the string is continued on the next line.

[1]Defined by the $LPDEF macro.

3.4 Line Printer Function Codes

The basic line printer I/O functions are write, sense mode, and set mode. None of the function codes take function modifiers.

3.4.1 Write

The line printer write functions print the contents of the user buffer on the designated printer.

The write functions and their QIO function codes are:

- IO$_WRITEVBLK—Write virtual block
- IO$_WRITELBLK—Write logical block
- IO$_WRITEPBLK—Write physical block (the data is not formatted, but output directly, as in PASSALL mode on terminals)

The write function codes can take the following device- or function-dependent arguments:

- P1—The starting virtual address of the buffer that is to be written
- P2—The number of bytes that are to be written
- P4—Carriage control specifier except for write physical block operations (write function carriage control is described in Section 3.4.1.1).

P3, P5, and P6 are not meaningful for line printer write operations.

In write virtual block and write logical block operations, the buffer specified by P1 and P2 is formatted for the selected line printer and includes the carriage control information specified by P4. The default buffer size is 132 bytes.

If the printer is not set spooled, write virtual block and write logical block operations perform the same function. If the printer is set spooled, a write logical block function queues the I/O to the printer, and a write virtual block function queues the I/O to the intermediate device, usually a disk.

All lowercase characters are converted to uppercase if the characteristics of the selected printer do not include LP$M_LOWER. (This does not apply to write physical block operations.)

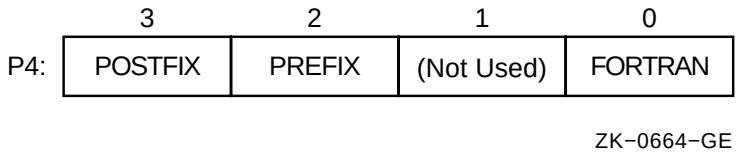
Multiple line feeds are generated for form feeds only if the printer does not have a mechanical form feed (LP$M_MECHFORM) characteristic. The number of line feeds generated depends on the current page position and the page length.

Section 3.2.1 describes character formatting in greater detail.

3.4.1.1 Write Function Carriage Control

The P4 argument is a longword that specifies carriage control. Carriage control determines the next printing position on the line printer. (P4 is ignored in a write physical block operation.) Figure 3–1 shows the P4 longword format.

Figure 3–1 P4 Carriage Control Specifier



ZK–0664–GE

Only bytes 0, 2, and 3 in the longword are used. Byte 1 is ignored. If the low-order byte (byte 0) is not 0, the contents of the longword are interpreted as a FORTRAN carriage control specifier. Table 3–3 lists the possible byte 0 values (in hexadecimal) and their meanings.

Table 3–3 Write Function Carriage Control (FORTRAN: byte 0 not equal to 0)

Byte 0 Value (hexadecimal)	ASCII Character	Meaning
20	(space)	Single-space carriage control (sequence: carriage-return/line-feed combination[1], print buffer contents, return)
30	0	Double-space carriage control (sequence: carriage-return/line-feed combination, carriage-return/line-feed combination, print buffer contents, return)
31	1	Page eject carriage control (sequence: form feed, print buffer contents, return)
2B	+	Overprint carriage control; allows double printing for emphasis or for special effects (sequence: print buffer contents, return)
24	$	Prompt carriage control (sequence: carriage-return/line-feed combination, print buffer contents)
All other values		Same as ASCII space character: single-space carriage control

[1]A carriage-return/line-feed combination is a carriage return followed by a line feed.

If the low-order byte (byte 0) is 0, bytes 2 and 3 of the P4 longword are interpreted as the prefix and postfix carriage control specifiers. The prefix (byte 2) specifies the carriage control before the buffer contents are printed. The postfix (byte 3) specifies the carriage control after the buffer contents are printed. The sequence is as follows:

1. Prefix carriage control
2. Print
3. Postfix carriage control

The prefix and postfix bytes, although interpreted separately, use the same encoding scheme. Table 3–4 shows this encoding scheme in hexadecimal format.

Table 3–4 Write Function Carriage Control (P4 byte 0 equal to 0)

Prefix/Postfix Bytes (Hexadecimal)		
Bit 7	**Bits 0–6**	**Meaning**
0	0	No carriage control is specified, that is, NULL.
0	1–7F	Bits 0 through 6 are a count of carriage-return/line-feed combinations.

(continued on next page)

Table 3–4 (Cont.) Write Function Carriage Control (P4 byte 0 equal to 0)

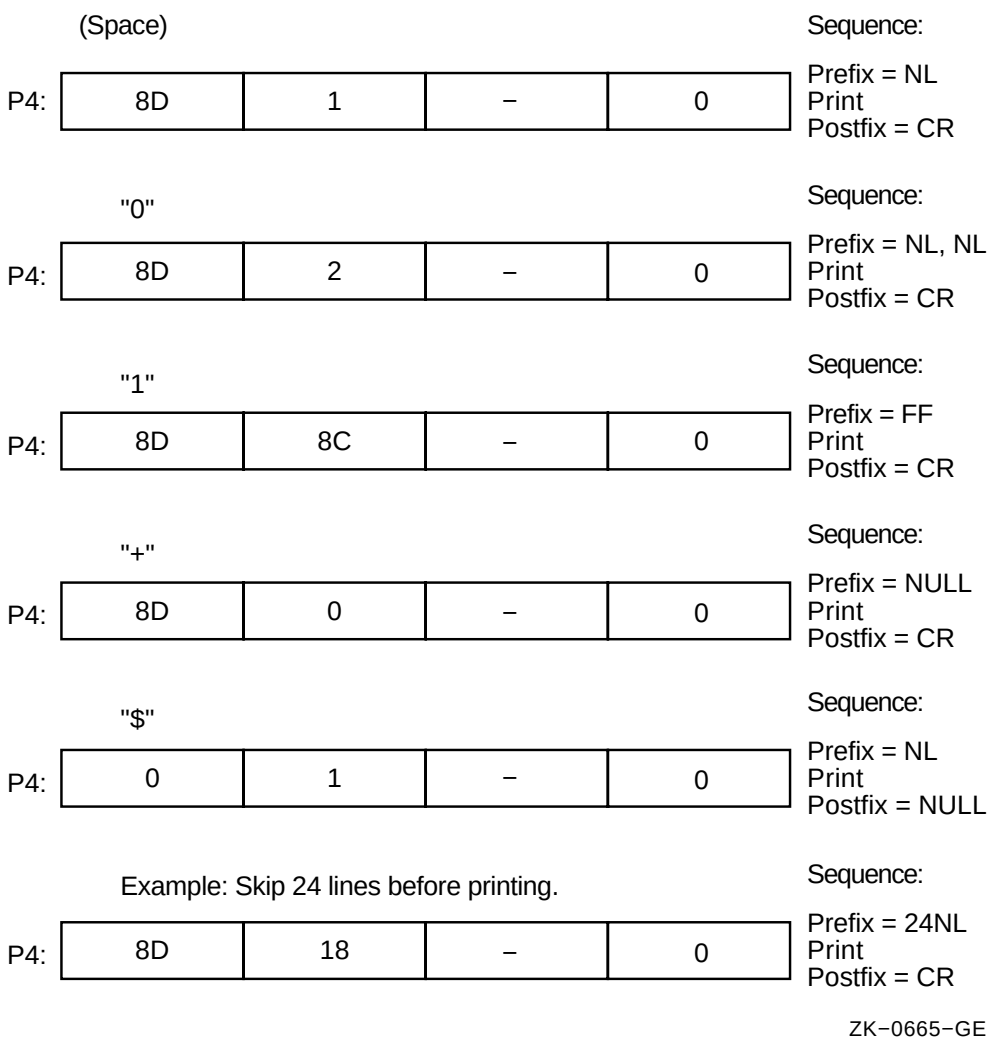
Bit 7	Bit 6	Bit 5	Bits 0–4	Meaning
1	0	0	1–1F	Output the single ASCII control character specified by the configuration of bits 0 through 4 (7-bit character set).
1	1	0	1–1F	Output the single ASCII control character specified by the configuration of bits 0 through 4, which are translated as ASCII characters 128 through 159 (8-bit character set; see the *OpenVMS I/O User's Reference Manual*).

Figure 3–2 shows the prefix and postfix hexadecimal coding that produces the carriage control functions listed in Table 3–3. Prefix and postfix coding provides an alternative way to achieve these controls.

In the first example, the prefix/postfix hexadecimal coding for a single-space carriage control (carriage-return/line-feed combination, print buffer contents, carriage-return) is obtained by placing the value (1) in the second (prefix) byte and the sum of the bit 7 value (80) and the return value (D) in the third (postfix) byte:

```
  80 (bit 7 = 1)
+  D (return)
 ----
  8D (postfix = return)
```

Figure 3–2 Write Function Carriage Control (Prefix and Postfix Coding)



3.4.2 Sense Printer Mode

The sense printer mode function senses the current device-dependent printer characteristics and returns them in the second longword of the I/O status block. No device- or function-dependent arguments are used with IO$_SENSEMODE.

3.4.3 Set Mode

Set mode operations affect the operation and characteristics of the associated line printer. The operating system provides two types of set mode functions: set mode and set characteristics. Set mode requires logical I/O privilege. Set characteristics requires physical I/O privilege. The following function codes are provided:

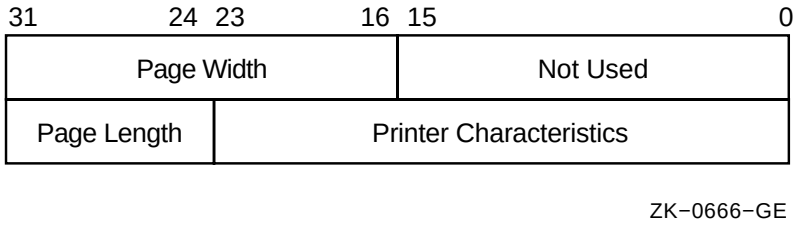
- IO$_SETMODE
- IO$_SETCHAR

These functions take the following device- or function-dependent argument (other arguments are not valid):

P1—The address of a characteristics buffer

Figure 3–3 shows the quadword P1 characteristics buffer for IO$_SETMODE.

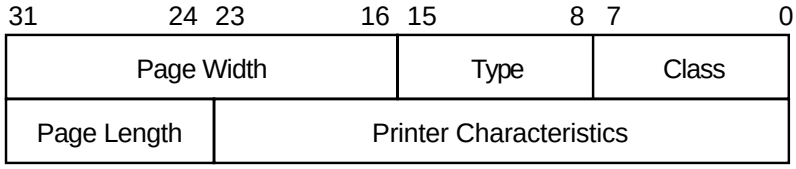
Figure 3–3 Set Mode Buffer



ZK–0666–GE

Figure 3–4 shows the same buffer for IO$_SETCHAR.

Figure 3–4 Set Characteristics Buffer



ZK–0667–GE

In the buffer, the device class is DC$_LP. The printer type is a value that corresponds to the printer: DT$_LP27 or DT$_LA11. The type can be changed by the IO$_SETCHAR function. The page width is a value in the range of 0 through 255 on a DMF32 controller and 0 through 65535 on an LP11 or DMB32 controller.

The printer characteristics part of the buffer can contain any of the values listed in Table 3–2.

Application programs that change specific line printer characteristics should perform the following steps:

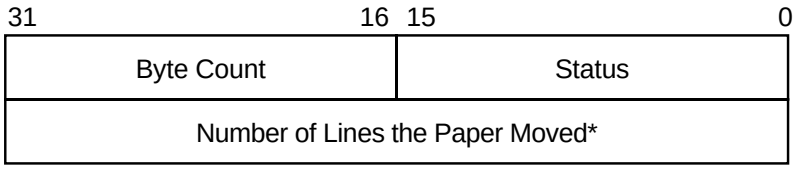
1. Use the IO$_SENSEMODE function to read the current characteristics.
2. Modify the characteristics.
3. Use the set mode function to write back the results.

Failure to follow this sequence will result in clearing any previously set characteristic.

3.5 I/O Status Block

The I/O status blocks (IOSB) for the write and set mode I/O functions are shown in Figures 3–5 and 3–6. Appendix A lists the status returns for these functions. (The OpenVMS system messages documentation provides explanations and suggested user actions for these returns.)

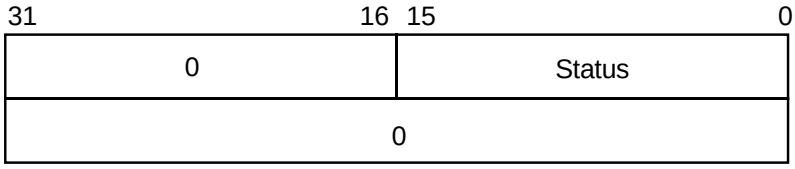
Figure 3–5 IOSB Contents — Write Function



* 0 if IO$_WRITEPBLK

ZK–0668–GE

Figure 3–6 IOSB Contents — Set Mode Function



ZK–0669–GE

3.6 Line Printer Driver Programming Example

The following sample program (Example 3–1) is an example of I/O to the line printer that shows how to use the different carriage control formats. This program prints out the contents of the output buffer (OUT_BUFFER) 10 times using 10 different carriage control formats. The formats are held in location OUTPUT_FORMAT.

Example 3–1 Line Printer Program Example

```
; ****************************************************************
;
        .TITLE  LINE PRINTER PROGRAMMING EXAMPLE
        .IDENT  /01/


;
; Define necessary symbols.
;
        $IODEF                          ;Define I/O function codes
;
; Allocate storage for the necessary data structures.
;
; Allocate output buffer and fill with required output text.
;
OUT_BUFFER:
        .ASCII  "VAX_PRINTER_EXAMPLE"
OUT_BUFFER_SIZE=.-OUT_BUFFER            ;Define size of output string
```

(continued on next page)

Example 3–1 (Cont.) Line Printer Program Example

```
;
; Allocate device name string and descriptor.
;
DEVICE_DESCR:                           ;
        .LONG   20$-10$                 ;Length of name string
        .LONG   10$                     ;Address of name string
10$:    .ASCII  /LINE_PRINTER/          ;Name string of output device
20$:                                    ;Reference label to calculate
;                                       ;length

;
; Allocate space to store assigned channel number.
;
DEVICE_CHANNEL:                         ;
        .BLKW   1                       ;Channel number

;
; Now set up the carriage control formats.
;
OUTPUT_FORMAT:                          ;
        .BYTE   0,0,0,0                 ;No carriage control
        .BYTE   32,0,0,0                ;Blank=LF+...TEXT...+CR
        .BYTE   48,0,0,0                ;Zero=LF+LF+...TEXT...+CR
        .BYTE   49,0,0,0                ;One=FF+...TEXT...+CR
        .BYTE   43,0,0,0                ;Plus=Overprint...+CR
        .BYTE   36,0,0,0                ;Dollar=LF+TEXT(Prompt)

;
; Now set up the prefix-postfix carriage control formats.
;
        .BYTE   0,0,1,141               ;LF+...TEXT...+CR
        .BYTE   0,0,24,141              ;24LF+...TEXT...+CR
        .BYTE   0,0,2,141               ;LF+LF+...TEXT...+CR
        .BYTE   0,0,140,141             ;FF+...TEXT...+CR
;
; ********************************************************************
;
;                       Start Program
;
; ********************************************************************

;
; The program assigns a channel to the output device, sets up a loop
; count for the number of times it wishes to print, and performs ten
; QIO and wait ($QIOW) system service requests.  The channel is then
; deassigned.
;
        .ENTRY  PRINTER_EXAMPLE,^M<R2,R3>  ;Program starting address
;
; First, assign a channel to the output device.
;
        $ASSIGN_S DEVNAM=DEVICE_DESCR,- ;Assign a channel to printer
                CHAN=DEVICE_CHANNEL     ;
        BLBC    R0,50$                  ;If low bit = 0, assign failure
        MOVL    #11,R3                  ;Set up loop count
        MOVAL   OUTPUT_FORMAT,R2        ;Set up o/p format address
                                        ;in R2
```

(continued on next page)

Example 3–1 (Cont.) Line Printer Program Example

```
;
; Start the printing loop.
;
30$:    $QIOW_S CHAN=DEVICE_CHANNEL,-   ;Print on device channel
                FUNC=#IO$_WRITEVBLK,-   ;I/O function is write virtual
                P1=OUT_BUFFER,-         ;Address of output buffer
                P2=#OUT_BUFFER_SIZE,-   ;Size of buffer to print
                P4=(R2)+                ;Format control in R2
                                        ;will autoincrement
        BLBC    R0,40$                  ;If low bit = 0, I/O failure
        SOBGTR  R3,30$                  ;Branch if not finished
40$:    $DASSGN_S CHAN=DEVICE_CHANNEL   ;Deassign channel
50$:    RET                             ;Return

        .END    PRINTER_EXAMPLE
```

A

I/O Function Codes

This appendix lists the function codes and function modifiers defined in the $IODEF macro. The arguments for these functions are also listed.

A.1 Card Reader Driver

Functions	Arguments	Modifiers
IO$_READLBLK IO$_READVBLK IO$_READPBLK	P1 - buffer address P2 - byte count	IO$M_BINARY IO$M_PACKED
IO$_SETMODE IO$_SETCHAR	P1 - characteristics buffer address	None
IO$_SENSEMODE	None	None

QIO Status Returns			
SS$_ABORT	SS$_DATAOVERUN	SS$_ENDOFFILE	SS$_NORMAL

A.2 Laboratory Peripheral Accelerator Driver

Functions	Arguments	Modifiers
IO$_LOADMCODE	P1 - starting address of microcode to be loaded P2 - load byte count P3 - starting microprogram address to receive microcode	None
IO$_STARTMPROC	None	None
IO$_INITIALIZE	P1 - address of initialize command table P2 - initialize command buffer length	None
IO$_SETCLOCK	P2 - mode of operation P3 - clock control and status P4 - real-time clock preset value (two's complement)	None

Functions	Arguments	Modifiers
IO$_STARTDATA	P1 - data transfer command table address P2 - data transfer command table length P3 - normal completion AST address P4 - overrun completion AST address	IO$_SETEVF

High-Level Language Subroutines	Functions
LPA$ADSWP	Start A/D converter sweep.
LPA$DASWP	Start D/A converter sweep.
LPA$DISWP	Start digital input sweep.
LPA$DOSWP	Start digital output sweep.
LPA$LAMSKS	Specify LPA11-K controller and digital mask words.
LPA$SETADC	Specify channel select parameters.
LPA$SETIBF	Specify buffer parameters.
LPA$STPSWP	Stop sweep.
LPA$CLOCKA	Set Clock A rate.
LPA$CLOCKB	Set Clock B rate.
LPA$XRATE	Compute clock rate and preset value.
LPA$IBFSTS	Return buffer status.
LPA$IGTBUF	Return next available buffer.

High-Level Language Subroutines	Functions
LPA$INXTBF	Alter buffer order.
LPA$IWTBUF	Return next buffer or wait.
LPA$RLSBUF	Release buffer to LPA11-K.
LPA$RMVBUF	Remove buffer from device queue.
LPA$CVADF	Convert A/D input to floating point.
LPA$FLT16	Convert unsigned integer to floating point.
LPA$LOADMC	Load microcode and initialize LPA11-K.

QIO Status Returns		
SS$_ABORT	SS$_BADPARAM	SS$_BUFNOTALIGN
SS$_CANCEL	SS$_CTRLERR	SS$_DATACHECK
SS$_DEVACTIVE	SS$_DEVCMDERR	SS$_DEVREQERR
SS$_EXQUOTA	SS$_INSFBUFDP	SS$_INSFMAPREQ
SS$_INSFMEM	SS$_IVBUFLEN	SS$_IVMODE
SS$_MCNOTVALID	SS$_PARITY	SS$_POWERFAIL
SS$_TIMEOUT		

A.3 Line Printer Driver

Functions	Arguments	Modifiers
IO$_WRITEVBLK IO$_WRITELBLK IO$_WRITEPBLK	P1 - buffer address P2 - buffer size P3 - (ignored) P4 - carriage control specifier[1]	None
IO$_SENSEMODE	None	None
IO$_SETMODE IO$_SETCHAR	P1 - characteristics buffer address	None

[1]Only for IO$_WRITEVBLK and IO$_WRITELBLK

QIO Status Returns			
SS$_ABORT	SS$_ACCVIO	SS$_CANCEL	SS$_NORMAL

Index

F

I

J

L

M

P

Q

R

S

T