pdp11

BASIC-11
Laboratory Extensions
User's Guide
Order No. DEC-11-LBEPA-A-D

digital

BASIC-11
Laboratory Extensions
User's Guide

Order No. DEC-11-LBEPA-A-D

digital equipment corporation · maynard, massachusetts

The postage prepaid READER'S COMMENTS form on the last page of this
document requests the user's critical evaluation to assist us in pre-
paring future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

CONTENTS (Cont.)

This manual describes the extensions for use with the BASIC/RT-11, BASIC/PTS-11, and BASIC/CAPS-11 systems. The extensions enable you to utilize PDP-11 lab peripherals such as LPS11, AR11, and DR11-K. Unless stated otherwise, the descriptions of all routines in this manual apply to the systems mentioned above.

The routines for these peripherals are provided in library form that can be linked with a user program. The user should have a programming knowledge of BASIC and some understanding of the peripherals.

The following table describes the documentation conventions used in this manual.

Convention	Meaning
<LF>	Line feed
<CR>	Carriage return
^ or CTRL	Used with special system control characters. Depress CTRL key while striking designated character.
ⓢ	Altmode
Square Brackets [ ]	Optional arguments are enclosed.
Lower case letter or lower case letter followed by a digit (a,b,x0,y1)	Value to be supplied by user. May be any valid arithmetic expression.
Lower case letter followed by a dollar sign (a$,x$)	String to be supplied by user. May be string constant (enclosed in quotes) or variable (A$).
Upper case letter (A,B,X,Y)	Numeric variable whose value will be determined by call or an array name.
Y axis	The vertical axis.
X axis	The horizontal axis.

The following manuals are necessary references for this manual:

RT-11 System Reference Manual
DEC-11-ORUGA-C-D

BASIC-11 Language Reference Manual
        DEC-11-LIBBA-B-D

BASIC/RT-11 Language Reference Manual
        DEC-11-LBACA-D-D

BASIC/PTS User's Manual
        DEC-11-LPTBA-A-D

BASIC/CAPS-11 User's Manual
        DEC-11-LIBCA-A-D

LPS11 User's Guide
        DEC-11-HLPGA-C-D

AR11 User's Guide
        DEC-11-HARUG

DR11-K Interface User's Guide and Maintenance Manual
        EK-DR11K-MM-001

CHAPTER 1

INTRODUCTION

BASIC Extensions support the RT-11, CAPS-11, and PTS-11 systems, and the following hardware:

    LPS11 Laboratory Peripheral System
    AR11 Analog Real-Time·Interface
    DR11-K Digital Input/Output Interface

The Laboratory Peripheral System (LPS) support can utilize LPS11, AR11 and DR11-K to sample and display data from analog to digital converters, digital input/output, or external events. LPS support contains 23 routines to control LPS11, AR11 and DR11-K. These 23 routines are divided into five categories according to their function. Each category is supplied as a module. The first module, LPS0, is the main module which contains all necessary support routines for using the other four modules. This module is required but the other four are optional.

The support for the peripherals consists of a library of routines that can be controlled by a user program through a CALL statement. The format of the CALL statement is:

        CALL "name"(argument list)

Under BASIC/CAPS-11, the routines can also be called by a statement of the form:

        name(argument list)

The function and limitation of each routine is described in detail in the following chapter.

CHAPTER 2

LABORATORY PERIPHERAL SYSTEM SUPPORT
(LPS11, AR11, DR11-K)

2.1 GENERAL FEATURES

Laboratory Peripheral System support for BASIC-11 allows a user to utilize the LPS hardware which includes LPS11, AR11 and up to 16 DR11-K. LPS support enables the sampling and displaying in a real-time environment of a variety of data from analog to digital converters, digital input/output, and external events. Sampling is controlled by crystal clocks and/or Schmitt triggers; it is possible to specify such parameters as sampling rate and response time thus allowing maximum flexibility.

NOTE

In a multiple DR11-K system, there must be a difference of 10(octal) between each unit in the interrupt and vector address. The status register address decreases by 10(octal) while the vector address increases by 10(octal).

All LPS routines are issued by the BASIC CALL statement allowing experienced PDP-11 assembly language programmers to easily include or modify the routines to meet particular (or special) requirements.

2.2 DESCRIPTION OF ROUTINES

The BASIC Extensions contain 23 routines to control the following options on the LPS11 hardware:

LPSAD-12	12-bit ADC, sample and hold, 8-channel multiplexer, and LED (light emitting diodes) 6-digit programmable decimal readout display.
LPSAD-NP	Direct memory access (DMA) option for the LPSAD-12 ADC.
LPSAG	Four differential preamplifiers with +or-1V input. Maximum of 4 LPSAGs per LPS11-S system.
LPSAG-VG	Four independently selectable multigain differential preamplifiers.
LPSAM	8-channel expansion multiplexer.
LPSSH	Second sample and hold for a dual sample and hold configuration.
LPSKW	Programmable real-time clock and two Schmitt triggers.
LPSVC	Display control including two 12-bit DACS. This

LPSDR   controller is capable of handling Digital's VR14 and VR20 scopes.

LPSDR   16-bit buffered digital I/O with drive capabilities and two programmable normally open (n.o.) relays. LPSDR cannot be used if DR11-K is used.

The 23 routines are divided into 5 categories according to their function. Each category is supplied as a separate module.

The following list is a summary of the routines available for controlling LPS hardware and a brief description of each:

MODULE LPS0 (This module is always required.)

USE	Defines array(s) to be used for storage of data.
ACC	Allows access to an entire array.
RDB	Returns the next data point from a specified buffer.

MODULE LPS1 (ANALOG TO DIGITAL CONVERSION)

ADC	Initiates an A/D conversion on a specified channel and returns the result to the user.
RTS	Performs real-time buffered/clocked sampling of the A/D.
LED	Displays a numeric value on the Light Emitting Diodes.
CVSG	Returns the next data value and gain in two separate variables.

MODULE LPS2 (REAL-TIME CLOCK)

SETR	Sets clock running at a designated rate and mode.
SETC	Sets clock running at a designated rate and initiates some action after a specified number of seconds have elapsed.
HIST	Performs histogram sampling using a timed Schmitt trigger.
WAIT	Waits for a specified event to occur.
RTIM	Returns the value of the internal software clock counter.

MODULE LPS3 (DIGITAL I/O)

DIR	Reads Digital Input register.
DOR	Writes Digital Output register.
DRS	Performs sampling of the Digital Input register.
REL	Closes or opens one of two relays.
IPK	Reads a value from a register or writes a value into a register.

MODULE LPS4 (DISPLAY)

CLRD	Defines display buffer and optionally clear or scale the data in it.
PUTD	Puts data into data buffer.

DIS	Displays data with incrementing x and variable y whenever BASIC is waiting for I/O.
FSH	Displays a single complete sweep of data with incrementing x and variable y.
DXY	Displays data with variable x and y values whenever BASIC is waiting for I/O.
FXY	Displays a single complete sweep of data with variable x and y values.

Module LPS0 is the main module and contains not only the USE, ACC, and RDB routines, but also all necessary support routines for the other modules. Therefore, it must be included, although the other modules are optional.

Data buffers used by the LPS routines differ from the normal arrays in BASIC in that they use only one word of storage per data element rather than two. This is because all LPS data is no larger than 2**16-1 and can be stored as unsigned 16-bit data. All data buffers must be defined by a USE routine before they are accessed by any other LPS routines. The USE routine allows the user to partition and make equivalent arrays for ease in displaying and manipulating common data. All data buffers defined in the USE routine are circular with internal pointers keeping track of where data is to be placed next and/or retrieved.

2.3   MODULE LPS0 (REQUIRED MODULE)

2.3.1   Defining the Buffer (USE)

```
USE
```

The USE routine defines buffer areas for use with the ACC, RDB, RTS, HIST, DRS, CLRD, PUTD, DIS, FSH, DXY and FXY routines. This routine sets up internal pointers allowing circular buffering and data overrun and/or nonexistent data checking. A maximum of five buffers may be specified, all of which must be given in a single USE statement. All areas defined in the USE statement must have been previously dimensioned in a DIM statement.

The format of the USE call is:

    CALL "USE"(A[(i)],[B[(j)],....,C[(f)]])

where

A,B,C	are the names of previously dimensioned array(s). May be 5 different arrays or array names may be repeated.
i,j,f	represents a valid subscript for the array or 0 which indicates the entire array.

The USE routine defines buffer areas required for storage of data. These areas may be a partitioned array which can be made equivalent to one large array. The following examples illustrate all aspects of the USE routine. Note that the size of an area defined in a DIM statement is one half that desired. This is because BASIC uses two words to store data whereas the LPS data is stored in one word.

Example:

Define areas A, B, and C to have 100, 200, and 300 data points respectively.

```
10 DIM A(50),B(100),C(150)
20 CALL "USE"(A,B,C)
```

Example:

Define area A to consist of three parts, the first having 10 data points and the second and third having 20 each. Then define a final area having access to all of the array A (including the zero subscript element).

NOTE

>Read the USE statement from left to right establishing the separate areas based on previously defined members of the same array. Only when the starting address of the next array is less than the previous one will entire access to the array be allowed by the following array.

The subscripts in the declaration are used to define pseudo partition names which can be used in other LPS statements which require arrays.

```
10 DIM A(25)
20 CALL "USE"(A(1),A(11),A(31),A)
```

In this example, each declaration of the array A has a unique description. A, which is equivalent to A(0), is different than A(1), A(11), and A(31). The following figure illustrates the partitioning of the array A due to the preceding example.
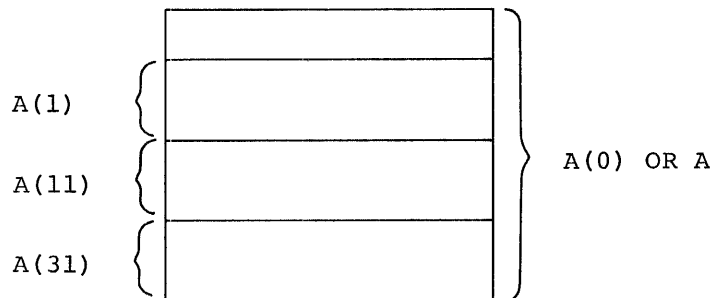
In the example, the partitioning occurs as follows: A(1) defines a buffer array starting at position A(1) in the array A and ending at the last position in the array. Since A(11) is declared immediately following it, the end of the array for A(1) is redefined to be one less than the A(11) position. At this point, A(1) goes from the 1st position in the array to the 10th, and A(11) goes from the 11th

position in the array to the 50th. When A(31) is declared immediately following A(11), A(11) is redefined to go from the 11th position in the array to the 30th and A(31) goes from the 31st position to the 50th. Now the partitioned array consists of three parts. The first part is called A(1) and is 10 locations in size. The second part is called A(11) and is 20 locations in size. The third part is called A(31) and is also 20 locations in size. The final declaration in the example is A or A(0) (both are equivalent), which allows access to the entire array A. This happens because the position in the array A of A(0) is less than the last declaration in the USE statement, i.e., A(31), and a new partitioning is started. This new partitioning begins at A(0) and proceeds until the end of the array A. The rules to continue from this point are the same as previously discussed and further partitioning could be defined if necessary. Note that every declaration in the USE statement must be unique, i.e., a statement of the form USE(A,A(0)) results in the first array A having an area of zero length. Since the second array is not unique in name, any reference to it later by other LPS statements actually refers to the array A and not A(0). Since A has zero length, the buffer declaration is useless.

2.3.2 Allowing Access to a Buffer (ACC)

```
 _____
|            |
|    ACC     |
|_____|
```

The ACC routine allows full access to the specified array by the RDB routine. The form of the call is:

        CALL "ACC"(A[(i)])

where

        A[(i)]   is a buffer previously declared in a call to the
                    USE routine.

A call to ACC resets all buffer pointers of the array A to allow full access to it by the RDB and PUTD routines. The PUTD pointer is placed at the end of the array and the RDB pointer is placed at the beginning. Since the PUTD pointer is placed at the end of the array, the buffer is considered full.

Example:

Allow full access to the array H and the buffer area of array A.

```
        10 DIM A(25),H(20)
        20 CALL "USE"(A(1),A(11),A(31),A,H)
        30 ....
        40 ....
         ..
         ..
         ..
        100 CALL "ACC"(H)
        110 CALL "ACC"(A(11))
        120 ....
        130 ....
         ..
         ..
         ..
```

2.3.3 Returning Data from a Buffer (RDB)

RDB

A call to the RDB routine returns the next data value from the specified buffer in the target variable. The value returned from RDB contains the data value with the gain in the most significant digits. The returned data is formatted. The form of the call is:

        CALL "RDB"(A[(i)],B)

where

        A[(i)]   is a buffer previously defined in a call to the
                    USE routine.
        B   is the target variable.

A call to the RDB routine returns values in B of 65535 to 0 for good data. A -1 will be returned if the data is bad (defined as software data overrun). A -2 will be returned if there is no data. A -3 will be returned if a hardware data overrun occurs.

A ring buffer is the specified buffer used by the RDB, RTS, HIST, PUTD, and DRS routines. Data is withdrawn from the ring buffer by the RDB routine but inserted into the ring buffer by the RTS, HIST, PUTD, or DRS routine. Two pointers which are invisible to the users point to the last data added and the next to be withdrawn.

If either pointer passes beyond the buffer, it is reset to the beginning of the buffer. If a RDB routine is executed when no data is available, the pointer is not advanced.

If a software data overrun occurs (that is, when there is no room in the ring buffer for data from the RTS, HIST, PUTD or DRS routine), then all subsequent calls on the RDB routine will return a -1. This will occur if the ring buffer is not large enough to contain all the data or the data is not removed from the ring buffer as fast as it is added.

When the referenced buffer contains analog sampling data (RTS function), the values returned in B are in the range 0 to 4095 for the LPS11 and 0 to 1023 for the AR11, with the gain in the most significant bits.

When the referenced buffer contains a clocked histogram sampling (HIST function), the values returned in B are in the range 0 to 65535. These values are either the number of ticks accumulated or the number remaining depending on the clock mode.

When the referenced buffer refers to digital I/O data (DRS function), a value between 0 to 65535 is returned in B from the next position in the specified buffer.

Example:

Assume that the array X has 100 data values previously entered by an RTS routine. Print out the data making sure that data overrun did not occur and that 100 data points were indeed taken.

```
10 FOR I=1 TO 100
20 CALL "RDB"(X,Z)
30 IF Z>=0 THEN 130
40 IF Z=-1 THEN 70
50 IF Z=-2 THEN 90
```

```
60 IF Z=-3 THEN 110
70 PRINT "RING BUFFER OVERRUN, SAMPLING TOO FAST";I
80 GO TO 150
90 PRINT "NO DATA AT EVENT YET";I
100 GO TO 140
110 PRINT "HARDWARE DATA OVERRUN, SAMPLING TOO FAST";I
120 GO TO 150
130 PRINT Z
140 NEXT I
150 CONTINUE
160 END

READY
```

2.4 MODULE LPS1 (A/D CONVERSION AND NUMERIC READOUTS)

2.4.1 Analog to Digital Conversion (ADC)

```
┌──────────┐
│   ADC    │
└──────────┘
```

The ADC routine initiates an A/D conversion from the specified channel, waits for it to complete, and returns the conversion as a floating point result in A (in the range 0 to 4095 for the LPS11 and 1023 for the AR11). The A/D cannot be currently involved in a real-time sampling (RTS) operation.

The form of the call is:

    CALL "ADC"(b,A[,i])
where
    b   specifies the channel and must be in the range
                from 0 to 63 for LPS11 and 0 to 15 for AR11.

    A   is the target variable and will contain the
                floating point conversion from the analog input.

                The value returned is as follows:

                    A=data/gain (where gain=1,4,16,64) For LPS11
                                          with gain.
                    A=data   When the AR11 is in
                                          bipolar mode. A is
                                          always positive.
                    A=-data   When the AR11 is in
                                          unipolar mode. A is
                                          always negative.

    i   specifies the gain setting for the requested A/D
                conversion on an LPS11, or the range (bipolar or
                unipolar) on an AR11.

                when

                i=0   software auto-gain ranging

                  =1   gain of 1 for LPS11 (-5V to +5V); bipolar
                        range for AR11 (-2.5V to +2.5V).

                  =2   gain of 4 for LPS11 (-1.25V to +1.24V);
                        unipolar range for AR11 (0 to +5V).

=3   gain of 16 for LPS11 (-0.3V to +0.3V);
illegal for AR11.

=4   gain of 64 for LPS11 (-0.078V to +0.078V);
illegal for AR11.

The default value is 1.

NOTE

On an LPS11, the LPSAG option must
be present when i is not equal to
1, otherwise the value returned in
the target variable will be
unspecified (i.e., dependent on
hardware).

Example:

Sample the A/D from channels 4 and 5, use a gain of 1, and save the
results in arrays A4 and A5 respectively. Assume 100 samples are to
be taken.

```
10 DIM A4(100),A5(100)
20 FOR I=1 TO 100
30 CALL "ADC"(4,A4(I))
40 CALL "ADC"(5,A5(I))
50 NEXT I
```

2.4.2 Real-Time Sampling (RTS)

```
RTS
```

The RTS routine performs real time buffered/clocked sampling of the
A/D.

The form of the call is:

CALL "RTS"(A[(i)],c1,c2,n,m[,i])

where

A[(i)]   is a buffer previously defined in a call to the
USE routine. The sampling will be stored in this
buffer. The data pointers in the ring buffer
will be reset before sampling is begun.

c1   determines the first channel to be sampled. Must
have a value between 0 and 15 for AR11 or LPS11
with gain other than 1 specified; or 0 to 63 for
LPS11 with no gain or gain equal to 1. This
value must be between 0 and 7 when dual sample
and hold is required.

c2   determines the number (or number of pairs) of
sequential channels which will be sampled. The
maximum is 16 for AR11 or LPS11 with gain other

than 1 specified; and 64 for LPS11 with no gain or gain equal to 1.

NOTE

The sum of C1 and C2 cannot be greater the 64.

n   determines the total number of samplings.

m   determines the mode of sampling.

i   specifies the gain setting for the requested A/D conversion.

when

i=0   software auto-gain ranging

=1   gain of 1 for LPS11 (-5V to +5V); bipolar range for AR11 (-2.5V to +2.5V).

=2   gain of 4 for LPS11 (-1.25V to +1.25V); unipolar range for AR11 (0 to +5V).

=3   gain of 16 for LPS11 (-0.3V to +0.3V); illegal for AR11.

=4   gain of 64 for LPS11 (-0.078V to +0.078V); illegal for AR11.

The default value is 1. When (c1 + c2) is greater than 16, it is illegal for i to be equal to zero or greater than 1.

NOTE

On an LPS11, the LPSAG option must be present when i is not equal to 1, otherwise the value returned in the target variable will be unspecified (i.e., dependent on hardware).

The A/D can be enabled in a variety of modes depending on the options specified. The normal mode of operation (m=0) causes the A/D to sample whenever Schmitt trigger 1 fires for the LPS11 or external start for the AR11. To enable other options, merely add their code number to the mode. The following table describes options available (all options are normally disabled):

Table 2-1
Mode Options in RTS Sampling

Code	Option
0	Normal operation, reads on Schmitt trigger 1 for LPS11 or external start for AR11.
+1	Enable burst mode (used only with DMA)
+2	Enable clock, disable Schmitt trigger 1 (used with LPSAD12, LPSKW or AR11)
+4	Enable dual sample and hold for LPS11 only (used with LPSSH)
+8	Enable DMA for LPS11 only (used with LPSAD-NP)

The burst mode can be used with Direct Memory Access only and the dual sample and hold mode can be used with the Direct Memory Access. The dual sample and hold, and DMA options can only be used on the LPS11. The following table lists all possible values for m and the modes enabled.

Table 2-2
Mode for Values of m

m		Mode
LPS11	AR11	
0	0	Schmitt trigger 1 or external start
2	2	Clock
4		Dual Sample and Schmitt trigger 1
6		Dual Sample and Clock
8		DMA and Schmitt trigger 1
9		DMA and Burst
10		DMA and Clock
11		DMA and Burst
12		DMA, Dual Sample, and Schmitt trigger 1
13		DMA, Dual Sample, and Burst
14		DMA, Dual Sample, and Clock
15		DMA, Dual Sample, and Burst

The A/D is started by a clock overflow or the firing of Schmitt trigger 1 (external start for AR11). Pointers are used to determine if data exists in the buffer arrays or if data wrap-around occurs. Since data is stored in circular buffers (excluding DMA operations), pointers are used to ensure that the incoming data rate does not exceed the removal rate. Data returned as -1 indicates ring buffer overrun; data returned as -2 indicates no data exist; data returned as -3 indicates hardware overrrun (interrupt service too slow). The buffer pointers are reset initially before the sampling operation begins.

A/D channels are sampled on every clock overflow or firing of Schmitt trigger 1 (external start for AR11) with the result stored in consecutive data cells. Data is stored in a format identical to that read from the A/D (bits 0-11 for data and bits 11-12 for gain). When a clock overflow or Schmitt trigger firing occurs, the A/D samples the first channel specified by c1 and then samples the next c2 minus 1 consecutive channels. Sampling then continues until n clock overflows or Schmitt triggers have been received. If n is specified as zero, any previous RTS sampling will be disabled.

In dual sample and hold mode, parameter c2 is the number of pairs of channels to read per sampling. Parameter n defines the number of samplings. There are 2*C2 elements per sampling.

DMA operations may use dual sample and hold. DMA allows direct hardware storage of A/D results from only one channel into a specified buffer array. A maximum of 4096 data may be taken at any one time with removal of data allowed only when the buffer is completely filled.

NOTE

When DMA is used with dual sample and hold, the parameter C2 must be 1, BASIC automatically assigns the value of 1 to C2 and any other number that is assigned to C2 will be ignored.

RTS operations do not interfere with sampling operations other than ADC (i.e., DRS and HIST routines) and all can be in progress simultaneously. You must set up the clock by making a call to the SETR routine before calling the RTS routine.

Example:

Set up the A/D to read data from channels 0 through 3 and store the results in the array A. Schmitt triggers are to be used to fire the A/D. Note that a dimension of 100 allows 200 data points. Since 4 channels are to be sampled, 50 Schmitt triggers will be required to complete the request.

```
10 DIM A(100)
20 CALL "USE"(A)
30 CALL "RTS"(A,0,4,100*2/4,0)
```

2.4.3 Display on the Light Emitting Diodes (LED)

```
┌─────────┐
│  LED    │
└─────────┘
```

A call to the LED routine displays the floating point value of the specified expression on the LEDs (Light Emitting Diodes) of the LPS11. Up to six positive or five negative digits can be displayed in the LEDs. An optional decimal point can also be included. Numbers which cannot be accurately displayed (i.e., E numbers or 6-digit negative numbers) are shown as all minus signs.

This routine acts as a NOP (no-operation) when $AR11 is defined in PERPAR.MAC for an AR11 system.

The form of the call is:

CALL "LED"(a)

where

a   the expression or the value to be displayed.

Example:

Display the value 5.632 on the LEDs.

```
10 A=5.632
20 CALL "LED"(A)
```

or

```
10 CALL "LED"(5.632)
```

2.4.4 Returning A/D Data from Buffer (CVSG)

```
 ┌────────────┐
 │  CVSG      │
 └────────────┘
```

The CVSG routine returns the unformatted data value sampled by the RTS routine in one variable and the gain in a separate variable. A -1 will be returned if the data is bad (defined as overrun). A -2 will be returned if there is no data. A -3 will be returned when a hardware data overrun occurs.

The form of the call is:

        CALL "CVSG"(V,I)

where

        V   is the target variable and contains the floating point conversion from the analog input in the range 0 to 4095 for LPS11 and 0 to 1023 for AR11.

        I   returns the gain setting of the requested A/D conversion.

                     when

                     =1   gain of 1 for LPS11 (+or-5V); bipolar range for AR11 (+or-2.5V).

                     =2   gain of 4 for LPS11 (+or-1.25V); unipolar range for AR11 (0 to +5V).

                     =3   gain of 16 for LPS11 (+or-.3V); illegal for AR11.

                     =4   gain of 64 for LPS11 (+or-.078V); illegal for AR11.

2.5 MODULE LPS2 (REAL-TIME CLOCK)

2.5.1 Setting the Clock Rate (SETR)

```
 ┌────────────┐
 │  SETR      │
 └────────────┘
```

A call to the SETR routine sets the clock running in the specified

mode and at the designated rate. The interrupt enable is always set (except mode 8 and above).

The form of the call is:

        CALL "SETR"(r,m,p)

where

r   determines the rate of the clock.

m   determines the mode of the clock.

p   is the preset value of the clock counter. The preset value must be less than 65535 (decimal) for the LPSKW and 255 (decimal) for the AR11.

The following preset values are illegal for the indicated rates in interrupt mode.

Rate   Illegal preset values

1   <150
                    <180 (for systems with memory management option)

2   <15
                    <18 (for system with memory management option)

3   =1

                            NOTE

            All values of rate and preset are legal in
            interrupt mode.

The following tables describe the rates and modes determined by the values of r and m.

Table 2-3
Selecting the Clock Rate

Values of r	Rate
0	No rate selected
1	1 MHz
2	100 kHz
3	10 kHz
4	1 kHz
5	100 Hz
6	Schmitt trigger 1 (external event for AR11)
7	Line frequency (50 Hz or 60 Hz)

2-13

Table 2-4
Selecting the Clock Mode

Values of m	Mode
0	Single interval mode. Counter counts from preset value to overflow and stops.
1	Repeated interval mode. Counter counts from preset value to overflow, transfers buffer/preset register to the counter, and begins again.
2	External event timing mode. The counter is free running, and a pulse from Schmitt trigger 2 transfers contents from the counter to the buffer/preset register and then continues counting. Not valid for AR11.
3	Event timing from zero base mode is the same as mode 2 except when the transfer of the counter to the buffer/preset register is done, the counter is cleared and the count begins from zero. Not valid for AR11.
4,5,6,7	Start clock only after Schmitt trigger 1 fires. Mode is then determined by the value of m-4. Not valid for AR11.
8 to 15	Indicates the operation of the clock in a non-interrupt mode has been added for faster A/D data acquisition. To get the value of m, just add 8 to the value of the interrupt mode. For example: m=0+8 indicates single interval non-interrupt mode clock operation. m=1+8 indicates repeated interval non-interrupt mode clock operation.

Example:

Set the clock running to interrupt once every second. A 100 Hz frequency is used and the clock mode is 1.

    CALL "SETR"(5,1,100)

Each programmable clock interrupt causes a 16-bit software clock (counter) to be incremented by one. When the maximum count of 65535 is reached, the next interrupt causes a reset to zero. This clock may be retrieved by the RTIM and DRS routines.

2.5.2 Setting the Clock to Rate and Time (SETC)

SETC

A call to the SETC routine sets clock to specified rate and time.

The form of the call is:

    CALL "SETC"(r,t)

where

r   determines the clock rate as described in
              Table 2-3, and may be 4,5, or 7.

t   is the time in seconds that the clock runs before
              issuing an interrupt.

The clock status register is set to rate determined by r and runs for
t seconds.   A clock interrupt then occurs which can be used to
initiate any of the clock controlled functions.   The time argument is
evaluated as ticks equal time in seconds multiplied by the clock rate
specified, e.g., if the clock rate was 10kHz, then ticks equal time in
seconds multiplied by 10kHz.   The ticks are entered into the clock
preset/buffer register.   The clock always runs in mode 0.

Example:

Set the clock to interrupt in 10 seconds using a 100 Hz frequency.

        CALL "SETC"(5,10)

2.5.3 Histograms - Timed Schmitt Trigger (HIST)

```
HIST
```

The HIST routine inputs values from the clock preset/buffer register
and stores them into the specified buffer whenever Schmitt trigger 2
fires.   The clock must be in mode 2 or 3 for the data to be
meaningful.

The form of the call is:

        CALL "HIST"(T[(i)],n)

where

        T[(i)]   is a buffer previously defined in a call to the
                  USE routine.

        n   determines the total number of data points stored.

The RDB function is used to retrieve the data.   The data pointers in
the ring buffer are reset before the sampling operation begins.

If n is given as zero, the HIST sampling will be disabled.

HIST operations do not interfere with other sampling operations (i.e.,
RTS and DRS) and all can be in progress simultaneously.

HIST routine acts as a NOP (no-operation) when $AR11 is defined in
PERPAR.MAC for an AR11 system.

Example:

Collect a timed histogram between external events (Schmitt trigger 2)
and store the results in array T.   The clock runs at 1 kHz and 100
intervals are required.

```
10 DIM T(50)
20 CALL "USE"(T)
30 CALL "HIST"(T,100)
40 CALL "SETR"(4,3,1)
```

2.5.4 Delaying Program Execution (WAIT)

WAIT

A call to WAIT disables further program execution until the specified event occurs.

The form of the call is:

        CALL "WAIT"(n)

where

>   n   specifies the event that must occur for program
>   execution to continue.
>
>   Values of n are:
>
>   n=0   Wait for clock to overflow only.
>   n=1   Wait for Schmitt trigger 1 (external event
>   for AR11) to fire (for clock rate = 6 only).
>   n=2   Wait for clock to overflow or Schmitt trigger
>   1 to fire.
>   n>2   Returns immediately.
>   n<0   Wait for Schmitt trigger 1 (external event
>   for AR11) to fire (a call to the SETR or SETC
>   routine must be made prior to this).

Example:

Wait for clock to overflow.

        10 CALL "WAIT"(0)

2.5.5 Returning Current Software Clock Tick Value (RTIM)

RTIM

A call to the RTIM routine returns the 16-bit integer value of the
internal software clock counter maintained by the programmable clock.

The form of the call is:

        CALL "RTIM"(s,t)

where

>   s   specifies whether the internal clock counter is to
>   be cleared or not. The counter is cleared when s
>   is equal to 0; otherwise it is unaltered.
>
>   t   is the 16-bit integer value of the internal clock
>   counter.

2.6 MODULE LPS3 (DIGITAL I/O)

The user should read the LPS11 Laboratory Peripheral System User's Guide in order to fully understand the hardware latching mechanism before using these modules.

The interrupt control logic permits the LPSDR-A or the DR11-K to perform an interrupt operation. The switches and jumpers on this logic can be arranged so that vector address can be assigned other than those configured as standard on the module for alignment.

One method of causing interrupts to the Unibus uses the two control lines between the DR11-K and the external device. If the input interrupt enable (bit 6 of the status register) is set, a negative transition (+3 v to ground) of the EXTERNAL DATA READY pulse will generate an interrupt to the Unibus, with a vector address of 300. A bus request is made on the BR level that corresponds with the level of the priority plug in the logic (the standard level for the DR11-K interface is BR4; this may be changed on the priority plug if desired). The control line method of interrupting is logically ORed into the DR11-K interrupt control, and is disabled by internal clamping circuitry if not desired. The device will continue interrupting as long as the line is held low.

The second and most preferred method of interrupting is to use the individual input lines. Each input (IN15:IN00) is buffered by a flip-flop that will set on a negative transition (+3 v to ground). Switches for the buffered bits on the hardware option make it possible to wire-OR each bit onto a common interrupt line. When the input interrupt enable (bit 6 of the status register) is set and a switch is on, the transition of the associated bit causes an interrupt to the Unibus. The bits are read under program control by reading the input register, and are cleared by moving data 1s to the bits to be cleared. The input interrupt enable is cleared when an input interrupt is accepted by the Unibus; when reset, it will retrigger the interrupt circuit if any other input bits were set during the program service subroutine, so that new interrupting bits will not be lost.

2.6.1 Reading the Digital Input Register (DIR)

DIR

A call to DIR reads the Digital Input Register and converts it to a floating point number. The form of the call is:

        CALL "DIR"(i,V,S[,m[,j]])

where

    i   determines the type of floating point conversion.

    V   is the target variable. It is the value in the input register ANDed with m.

    S   contains the returned digital Control Status Register (CSR) setting.

m	indicates the 16-bit mask. The default value is -1.
j	indicates the number of the DR11-K unit on a multiple DR11-K system. The default value is 0. The maximum legal value is determined by the parameter $NUMBER in PERPAR.MAC at the assembly time of PERVEC.MAC module.

If i=0, input is four Binary Coded Decimal (BCD) digits converted to a floating point number and the result is in the range 0 to 9999. If i<>0, then the binary result read from the register is directly converted to a floating point number and the result is in the range 0 to 65535. The Digital Input Register is read via an internal load request and does not respond to interrupts. The input word is immediately written back into the input register to clear those bits which were obtained from the register. The result is returned in V.

The new CSR register setting is returned in S.

Example:

```
1 REM THIS PROGRAM TESTS THE "DIR","DOR" AND "DRS" MODULES ALONG WITH
2 REM SETR,WAIT AND RDB. THE INPUT AND THE OUTPUT DIGITAL CABLES MUST BE
3 REM CONNECTED TO EACH OTHER.
4 DIM X(30)
5 O=1
6 CALL "DOR"(1,65535,N)
7 CALL "USE"(X)
10 FOR I=1 TO 16
20 CALL "DOR"(O,O,N)
30 CALL "DIR"(1,Y,N1)
40 PRINT N,Y
50 O=O*2
60 NEXT I
160 CALL "DOR"(1,65535,N)
165 CALL "DRS"(X,1,30,0,N)
170 CALL "SETR"(5,1,100)
175 M=0
180 O=0
185 FOR I=1 TO 30
190 CALL "DOR"(1,65535,N)
195 CALL "DOR"(M,O,N)
200 O=O+4096+256+16+1
205 CALL "WAIT"(O)
210 NEXT I
215 GOSUB 300
220 STOP
300 FOR J=1 TO 30
305 CALL "RDB"(X,Y)
310 PRINT J,Y
315 NEXT J
320 RETURN
500 END
```

2.6.2 Writing to the Digital Output Register (DOR)

```
┌─────────┐
│   DOR   │
└─────────┘
```

A call to DOR can either set or clear selected bits in the Digital Output Register. The form of the call is

        CALL "DOR"(m,n,R[,j])

where

m	determines whether bits are to be set (when m=0) or cleared (when m is not equal to 0).
n	determines which bits are to be set or cleared.
R	contains the floating point equivalent to the new value in the Digital Output Register.
j	indicates the number of the DR11-K unit on a multiple DR11-K system. The default value is 0. The maximum legal value is determined by the parameter $NUMBER in PERPAR.MAC at the assembly time of PERVEC.MAC module.

If a bit in the binary representation of n is 1, the corresponding bit of the Digital Output Register will be cleared or set (depending on the value of m). If a bit in the binary representation of n is 0, the corresponding bit of the Digital Output Register will not be changed. The BASIC-11 BIN and OCT functions are very useful in setting or clearing the registers.

Example:

Turn on (set) bits 1 and 2 of the Digital Output Register.

        40 CALL "DOR"(0,BIN'110',N) Clear Digital Output Register.

        40 CALL "DOR"(1,OCT'177777',N)

                or

        40 CALL "DOR"(1,-1,N)

2.6.3 Digital Readout Sampling (DRS)

```
┌─────────┐
│   DRS   │
└─────────┘
```

A call to the DRS routine samples the Digital Input Register in a similar fashion as the RTS function.

The form of the call is:

        CALL "DRS"(A[(i)],m1,n,m2,R[,T[,m[,j]]])

where

A[(i)]   is a buffer previously defined in a call to the USE routine.

m1   determines the mode by which the Digital Input Register is to be read.

n   determines the total number of samplings.

m2   determines whether the sampling is clock driven.

R   contains the returned setting of the digital Control Status Register (CSR).

T   describes the address of the buffer to store the current clock TICK value on every data interrupt when m2 is not equal to zero.

m   indicates the 16-bit mask. The default value is -1.

j   indicates the number of the DR11-K unit on a multiple DR11-K system. The default value is 0. The maximum legal value is determined by the parameter $NUMBER in PERPAR.MAC at the assembly time of PERVEC.MAC module.

When m2 is equal to 0, each time the clock fires (Schmitt trigger, or external event for AR11), the Digital Input Register is read.

If m1 is equal to zero the Digital Input Register will be treated as Binary Coded Decimal and will be converted to binary. If m1 is not equal to zero the Digital Input Register will be input directly as a binary number. This number is stored in the circular buffer specified by A[(i)]. When DRS is first called, it resets the pointers of the buffer to the beginning.

If n is given as zero, the DRS sampling will be disabled.   The DRS call is driven by digital clock when m2 is not equal to zero. Whenever a new value is received in the input register, the value is immediately read in and stored in the buffer. The value of the 16-bit software clock is stored in the buffer specified by T.   The input data word is immediately written back into the input register, and the active bit which have been sampled are cleared.

The new setting of the digital Control Status Register is returned in R.

Example:

Read the Digital Input Register once every one tenth of a second for 100 readings and store the results in array A.

```
10 DIM A(50)
20 CALL "USE"(A)
30 CALL "DRS"(A,0,100,0,N)
40 CALL "SETR"(5,1,10)
```

2.6.4 Relay Control (REL)

```
 REL 
```

A call to the REL routine opens or closes the specified relay.

The REL routine acts as a NOP (no-operation) when $AR11 is defined in PERPAR.MAC for an AR11 system.

The form of the call is:

        CALL "REL"(a,b)

where

      a   specifies the relay and may be equal to 1 or 2.

      b   determines the operation.   Relay is opened if equal to 0, otherwise it is closed.

Example:

Open relay 1 and close relay 2.

```
100 CALL "REL"(1,0)
110 CALL "REL"(2,1)
```

2.6.5 Reading/Writing Data From/Into Memory (IPK)

```
 IPK 
```

The IPK routine can be called to read a value from a specified address or place a value into a specified address. The form of the call is:

        CALL "IPK"(s,a,V)

where

      s   indicates whether the value is a word or a byte. If s is even, the word value supplied in V will be read from or written into the even address register. If s is odd, the byte value supplied in V will be read from or written into the odd address register.

                A call such as CALL "IPK" (0,64,V) reads into variable V the value at location 64 in memory.

      a   specifies the address where the value V is read from or written into. This address must be even when the value of s is even. This address can be specified as an octal string, integer constant or integer variable.

                A call such as CALL "IPK" (2,73,100) is illegal because the address is odd; the error message ?ARG will result.

V   is the value to be read from or written   into   the
                 address register.

                                  NOTE

                 This routine should not be used except   to
                 read from or write into the I/O page or to
                 read from memory. Writing into memory can
                 cause serious consequences (program being
                 wiped out, etc.).

2.7 MODULE LPS4 (DISPLAY)

The routines in this module require the LPSVC or AR11 with the VR14
interfaced through it. The VT11 cannot be used with these routines.

2.7.1 Defining the Display Buffer (CLRD)

CLRD

A call to the CLRD routine defines the display buffer having fixed
delta x values.

The form of the call is:

        CALL "CLRD"(A[(i)],a,b)

where

     A[(i)]   is a buffer previously defined by a call to the
                     USE routine.

     a   specifies the size of the buffer to be displayed.

     b   specifies the scale.

The buffer to be displayed should contain single word values.   Values
in the range 4095>=value>=0 are displayed while values outside this
are not. The size of the buffer, a, is the   number   of   Y   points   to
display and must be less than or equal to the number of points defined
in the DIM statement and the call to the USE routine. The delta x is
calculated as 4096/a and can be fractional.

If b, the scale, equals 0, CLRD will set all buffer values to -1
(non-displayable values).   If scale does not equal 0, CLRD bypasses
the clearing of the array and the original data is multiplied by b.
In either case, the PUTD pointers are reset to point to the beginning
of the array. Data is entered into the array through the PUTD
function; however, a CLRD must be issued before data is initially
transferred to the array.

A CLRD routine must be issued at least once before issuing the DIS,
FSH, or DXY functions which can display the buffer defined by CLRD.

Example:

Set up the array C to be used as a display buffer having 256 points.

```
10 DIM C(128)
20 CALL "USE"(C)
30 CALL "CLRD"(C,256,0)
```

2.7.2 Putting Data into Display Buffer (PUTD)

```
┌──────────┐
│  PUTD    │
└──────────┘
```

A call to the PUTD routine puts a data value into the specified buffer. Repeated calls to PUTD will cause the buffer to be filled sequentially.

The form of the call is:

        CALL "PUTD"(A[(i)],b)

where

A[(i)]   is a buffer previously defined by the USE routine.

b   is the value to be inserted. Must be in range $0<=b<=65535$.

This function does not initiate a display, but rather just enters data into the specified array.

Example:

Remove 100 data points from the specified digital sampling buffer D, and transfer them to the buffer Z.

```
80 DIM D(50),X(50)
90 CALL "USE"(D,X)
100 FOR I=1 TO 100
110 CALL "RDB"(D,X)
120 CALL "PUTD"(Z,X)
130 NEXT I
```

2.7.3 Background Display Routine (DIS)

```
┌──────────┐
│  DIS     │
└──────────┘
```

A call to the DIS routine displays data from the buffer whenever BASIC is idle. Data is not displayed by DIS or DXY routines under RT-11 FB system because BASIC is never idle. Data is displayed under RT-11 SJ system (version 2 or later) only when I/O is taking place.

The form of the call is:

        CALL "DIS"(A[(i)],a,b[,n]) where

A[(i)]   is a display buffer previously defined by the USE and CLRD routines.

a   determines the starting point of the display.

b   determines the frequency of points in the buffer
                that are to be displayed.

n   specifies the number of data point to display.
                The default value is all remaining elements of the
                array.

The points displayed start with the point a in the buffer and proceed
in increments of b. If b is equal to 1, consecutive points starting
with the a one are displayed. If b is equal to 2, every other point
is displayed, etc.

Example:

Display data from buffer E beginning at the 12th data point and
displaying every 3rd point of the remaining array elements.

```
20 DIM E(100)
40 CALL "USE"(E)
60 CALL "CLRD"(E,200,0)
80 REM BUFFER MAY BE FILLED HERE
100 CALL "DIS"(E,12,3)
120 REM OR MAY BE FILLED HERE
```

Example:

```
5 REM THIS PROGRAM TESTS THE ROUTINES-USE,CLRD,PUTD,DIS,FSH-ALONG WITH
10 REM SETR,AND WAIT
15 PRINT "PROGRAM TO TEST CLRD,PUTD,DIS AND FSH."
50 CALL "SETR"(5,1,10)
100 DIM A(100)
200 CALL "USE"(A)
300 CALL "CLRD"(A,100,0)
400 FOR I=1 TO 100
500 CALL "PUTD"(A,I)
600 NEXT I
700 CALL "DIS"(A,1,1)
710 INPUT D
750 CALL "DIS"(A,1,1,75)
760 INPUT D
780 CALL "DIS"(A,1,1,50)
800 INPUT D
900 IF D=0GO TO 2000
950 FOR S=1 TO 99
975 FOR I=1 TO 50
980 FOR D=1 TO 100
1000 CALL "FSH"(A,S,I,D)
1010 CALL "WAIT"(0)
1025 NEXT D
1030 NEXT I
1050 NEXT S
1100 GO TO 800
2000 STOP
```

2.7.4 Display Buffer (FSH)

```
┌─────────┐
│  FSH    │
└─────────┘
```

The FSH routine is identical to the DIS routine except that the data
points in the buffer are completely displayed only once when this call
is executed. The next BASIC statement is then executed.

The form of the call is:

        CALL "FSH"(A[(i)],a,b[,n])

where

        A[(i)]   have the
        a   same
        b   meaning
        n   as in DIS.

Example:

Using the previous example, display 100 cycles of the array E.

        100 FOR I=1 TO 100
        110 CALL "FSH"(E,12,3)
        120 NEXT I

2.7.5 Displaying X,Y Data (DXY)

```
┌─────────┐
│  DXY    │
└─────────┘
```

A call to the DXY routine displays points from two buffers as x and y
values.   These buffers are displayed whenever BASIC is idle. Data is
displayed under RT-11 SJ system (version 2 or later) only when I/O is
taking place.

The form of the call is:

        CALL "DXY" (X[(i)],Y[(j)],a,b[,n])

where

        X[(i)]   is a buffer previously defined by a call to the
                        USE routine and contains the x values.

        Y[(j)]   is a buffer previously defined by calls to the USE
                        and CLRD routines and contains the y values.

        a   determines the starting point in both buffers

        b   determines the frequency of the points in each
                        buffer to be displayed.

        n   specifies the number of data points to display.
                        The default value is all remaining elements of the
                        array.

The buffer containing the x values, X[(i)] does not have to be initialized by a call to CLRD, but it may be convenient to do so to initialize all values so that they are non-displayable. The buffer containing the y values must be initialized by a call to CLRD although the value of delta x is not used.

As in the other display routines the a determines the location of the first point to be displayed in each buffer and b determines the frequency of points to be displayed. If b equals one, consecutive values are taken from the two buffers. If b equals two, every other value in each buffer will be used to create the display.

Example:

Generate fiducial marks on the display screen of a 256-point display every 16 points. Marks will be 10 points in height. Data will be generated into the arrays X and Y.

```
5 REM THIS PROGRAM TESTS -DXY AND FXY- ROUTINES
10 PRINT "TEST PROGRAM FOR DXY AND FXY ROUTINES"
20 DIM X(128),Y(128)
30 CALL "USE"(X,Y)
40 CALL "CLRD"(X,256,0)
50 CALL "CLRD"(Y,256,0)
60 FOR I=16 TO 256 STEP 16
70 FOR J=1 TO 10
80 CALL "PUTD"(X,I)
90 CALL "PUTD"(Y,J)
100 NEXT J
110 NEXT I
120 CALL "DXY"(X,Y,1,1)
130 STOP
140 END

READY
```

2.7.6 Flashing X-Y Data (FXY)

```
┌─────────┐
│  FXY    │
└─────────┘
```

The FXY routine is the same as DXY routine except that the X and Y values are displayed only once when this call is made.

The form of the call is:

        CALL "FXY"(X[(i)],Y[(j)],a,b[,n])

X[(i)]	Have the
Y[(j)]	same
a	meaning
b	as in
n	DXY.

Example:

```
5 REM THIS PROGRAM TESTS -DXY AND FXY- ROUTINES
10 PRINT "TEST PROGRAM FOR DXY AND FXY ROUTINES"
20 DIM X(128),Y(128)
30 CALL "USE"(X,Y)
40 CALL "CLRD"(X,256,0)
50 CALL "CLRD"(Y,256,0)
```

```
60 FOR I=16 TO 256 STEP 16
70 FOR J=1 TO 10
80 CALL "PUTD"(X,I)
90 CALL "PUTD"(Y,J)
100 NEXT J
110 NEXT I
120 CALL "DXY"(X,Y,1,1)
130 INPUT D
140 CALL "DXY"(X,Y,1,1,25)
150 INPUT D
160 CALL "DXY"(X,Y,1,1,50)
170 INPUT D
180 IF D=0GO TO 260
190 FOR S=1 TO 99
200 FOR I=1 TO 50
210 FOR D=1 TO 100
220 CALL "FXY"(X,Y,S,I,D)
230 NEXT D
240 NEXT I
250 NEXT S
260 STOP
270 END

READY
```

2.8 HARDWARE REQUIRED FOR LPS COMMANDS

The following summary describes the hardware necessary to fully utilize the LPS system.

Command	Hardware Required
USE	None
ACC	None
RDB	None
ADC	LPSAD-12, LPSAM (for additional 8 channels)
RTS	LPSAD-12, LPSAD-NP (for DMA operations)
	LPSAM (for additional 8 channels)
	LPSSH (for dual sample and hold)
	LPSKW (for real-time clocking and Schmitt triggers)
	LPSAG-VG (for multi-gain)
LED	LPSAD-12
CVSG	None
SETR	LPSKW
SETC	LPSKW
HIST	LPSKW
WAIT	LPSKW
RTIM	LPSKW
DIR	LPSDR
DOR	LPSDR
DRS	LPSDR
REL	LPSDR
IPK	None
CLRD	None
PUTD	None
DIS	LPSVC
FSH	LPSVC
DXY	LPSVC
FXY	LPSVC

The routines that require the AR11 are ADC, RTS, SETR, SETC, WAIT, and RTIM.

The routines that require the VR14 in addition to the AR11 are DIS, FSH, DXY, and FXY.

The routines that require the DR11-K are DIR, DOR, and DRS.

2.9 EXAMPLE PROGRAMS

```
1 REM    THIS PROGRAM TESTS THE "ADC" MODULE
2 REM    IT REQUIRES 2 INPUTS: CHANNEL # AND IGAIN FROM CONSOLE
3 REM    A CHANNEL # EQUAL TO -1 TERMINATES PROGRAM
10 INPUT C
20 INPUT G
30 IF C=-1 THEN 100
40 CALL "ADC"(C,V)
50 PRINT V,ABS(V)
60 CALL "ADC"(C,V1,1)
70 PRINT V1,ABS(V1)
80 CALL "ADC"(C,V2,G)
90 PRINT V2,ABS(V2)
95 GO TO 10
100 END

**

1 REM THIS PROGRAM TESTS THE DRS MODULE TIME LOG FEATURE
3 DIM Z(30)
4 DIM X(30)
5 O=1
6 CALL "DOR"(1,65535,N)
8 CALL "USE"(X,Z)
10 FOR I=1 TO 16
20 CALL "DOR"(0,0,N)
30 CALL "DIR"(1,Y,N1)
40 PRINT N;Y
50 O=O*2
60 NEXT I
160 CALL "DOR"(1,65535,N)
165 CALL "DRS"(X,1,30,1,N,Z,7)
166 PRINT "********",N,"***********"
170 CALL "SETR"(4,1,1)
175 M=0
180 O=0
185 FOR I=1 TO 30
190 CALL "DOR"(1,65535,N)
195 CALL "DOR"(M,O,N)
200 O=O+256+16+1
210 NEXT I
215 GOSUB 300
220 STOP
300 FOR J=1 TO 30
305 CALL "RDB"(X,Y)
306 CALL "RDB"(Z,Y1)
310 PRINT J,Y,Y1
315 NEXT J
320 RETURN
500 END
*
10 REM    THIS PROGRAM TESTS THE "RTS" MODULE
20 REM    IT REQUIRES 3 INPUTS: STARTING CHANNEL #, NUMBER OF CHANNELS
30 REM    AND IGAIN FROM CONSOLE
40 REM    A STARTING CHANNEL # EQUAL TO -1 TERMINATES PROGRAM
50 DIM A(20)
```

```
60 CALL "USE"(A)
70 INPUT S
80 IF S=-1 THEN 150
90 INPUT N
100 INPUT G
110 CALL "SETR"(5,1,100)
120 CALL "RTS"(A,S,N,20,2,G)
130 GOSUB 200
140 GO TO 50
150 STOP
160 REM@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
170 CALL "RDB"(A,Z)
180 IF Z<0 THEN 170
190 RETURN
200 FOR I=1 TO 4
210 GOSUB 170
220 A1=Z
230 GOSUB 170
240 A2=Z
250 GOSUB 170
260 A3=Z
270 GOSUB 170
280 A4=Z
290 GOSUB 170
300 A5=Z
310 PRINT A1,A2,A3,A4,A5
320 NEXT I
330 RETURN
340 END

READY
*
1 REM     THIS PROGRAM TESTS THE "RTS" & "CVSG" ROUTINES
2 REM     IT REQUIRES 3 INPUTS: STARTING CHANNEL #, NUMBER OF CHANNELS
3 REM     AND IGAIN FROM CONSOLE
4 REM     A STARTING CHANNEL # EQUAL TO -1 TERMINATES PROGRAM
10 DIM A(20)
20 CALL "USE"(A)
30 INPUT S
40 IF S=-1 THEN 100
50 INPUT N
60 INPUT G
70 CALL "SETR"(5,1,100)
80 CALL "RTS"(A,S,N,20,2,G)
90 GOSUB 220
95 GO TO 10
100 STOP
101 REM@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
200 CALL "CVSG"(Z,V)
210 IF Z=-2 THEN 200
215 RETURN
220 FOR I=1 TO 4
225 GOSUB 200
230 A1=Z
235 V1=V
240 GOSUB 200
250 A2=Z
255 V2=V
260 GOSUB 200
270 A3=Z
275 V3=V
280 GOSUB 200
290 A4=Z
```

```
295 V4=V
300 GOSUB 200
310 A5=Z
315 V5=V
320 PRINT A1,A2,A3,A4,A5
325 PRINT V1,V2,V3,V4,V5
350 NEXT I
400 RETURN
1000 END
*
1 REM THIS PROGRAM TESTS THE "RTIM" MODULE ALONG WITH SETR AND WAIT
2 REM THE LAST 10 VALUES PRINTED WILL BE 0 THRO' 9.
10 INPUT S
20 IF S=-1 THEN 1000
30 CALL "RTIM"(0,V)
40 PRINT V
50 CALL "SETR"(5,1,100)
60 FOR I=1 TO 10
70 CALL "RTIM"(1,V)
80 PRINT V
90 CALL "WAIT"(0)
100 NEXT I
110 GO TO 10
1000 STOP

*
```

COMMAND SUMMARY

The following list is a summary of the routines available for controlling LPS hardware and a brief description of each:

CALL	ARGUMENT LIST	EFFECT
USE	(A[(i)][B[(j)]...C[(f)]])	Defines array(s) to be used for storage of data.
ACC	(A[(i)])	Allows access to an entire array.
RDB	(A[(i)], B]	Returns the next data point from a specified buffer.
ADC	(b,A[,i])	Initiates an A/D conversion on a specified channel and return the result to the user.
RTS	(A[(i)], c1,c2,n,m[,i])	Performs real-time buffered/clocked sampling of the A/D.
LED	(a)	Displays a numeric value on the Light Emitting Diodes.
CVSG	(V,I)	Returns the next data value and gain in two separate variables.
SETR	(r,m,p)	Sets clock running at a designated rate and mode.
SETC	(r,t)	Sets clock running at a designated rate and initiates some action after a specified number of seconds have elapsed.
HIST	(T[(i)],n)	Performs histogram sampling using a timed Schmitt trigger.
WAIT	(n)	Waits for a specified event to occur.
RTIM	(s,t)	Returns the value of the internal software clock counter.
DIR	(i,V,S[,m[,j]])	Reads Digital Input register.
DOR	(m,n,R[,j])	Writes Digital Output register.

CALL	ARGUMENT LIST	EFFECT
DRS	(A[(i)],m1,n2,m2, R[,T[,m[,j]]])	Performs sampling of the Digital Input register.
REL	(a,b)	Closes or opens one of two relays.
IPK	(s,a,V)	Reads a value from a register or writes a value into a register.
CLRD	(A[(i)],a,b)	Defines display buffer and optionally clears or scales the data in it.
PUTD	(A[(i)],b)	Puts data into data buffer.
DIS	(A[(i)], a,b[,n])	Displays data with constant x and variable y whenever BASIC is waiting for I/O.
FSH	(A[(i)],a,b[,n])	Displays a complete sweep of data with constant x and variable y.
DXY	(X[(i)],Y[(j)],a,b[,n])	Displays data with variable x and y values whenever BASIC is waiting for I/O.
FXY	(X[(i)],Y[(j)],a,b[,n])	Displays data with variable X and Y values respectively only once.

APPENDIX B

BUILDING LOAD MODULES

B.1 BASIC/RT-11

B.1.1 LPS Support

The software supplied supports the standard hardware configuration
only. The standard hardware configuration is determined by the
address of the device hardware register and vectors. The standard
LPS11 or AR11 register is at 170400 (octal) and the vector is at 340
(octal). The standard DR11-K register is at 167770 (octal) and vector
is at 300 (octal).

The supplied software has to be reassembled and relinked if the
hardware configuration does not correspond to the above. To redefine
the hardware register and vector address, just edit the source file
PERPAR.MAC.

The LPS Support for BASIC/RT-11 is supplied in ten binary relocatable
files (on DECpack disk, DECtape, floppy disk, or 9-track magtape).
These files are:

LPS0.OBJ	Required	LPS kernel module for LPS11 or AR11 and DR11-K
LPS1.OBJ	Optional	Analog to digital conversion for LPS11
ARD1.OBJ	Optional	Analog to digital conversion for AR11 and DR11-K
LPS2.OBJ		Real-time clock (60 Hz line frequency) for LPS11
ARD2.OBJ		Real-time clock for AR11 (60 Hz line frequency) and DR11-K
LPS2C.OBJ	One is required	Real-time clock (50 Hz line frequency) for LPS11
ARD2C.OBJ		Real-time clock (50 Hz line frequency) for AR11 and DR11-K
LPS3.OBJ	Optional	Digital input/output for LPS11
ARD3.OBJ	Optional	Digital input/output for AR11 and DR11-K
LPS4.OBJ	Optional	Display for LPS11 or AR11 and DR11-K

The following files are also provided in source form in all kits:

FTBL.MAC	Function Table Module
PERVEC.MAC	Vector Definition Module
RTINT.MAC	Interface Module
PERPAR.MAC	Parameter

To build a load module BASLPS.SAV (BASIC with LPS11 support) or

BASARD.SAV (BASIC with AR11 and DR11-K support), first transfer all
BASIC Extensions binaries (including sources) and BASIC files to the
system device with PIP. The parameter file PERPAR.MAC is then edited
and assembled with FTBL.MAC, PERVEC.MAC, and RTINT.MAC. The three
object modules produced are then linked with the LPS and BASIC object
modules to produce a load module.

NOTE

All of the procedures in this section
assume that an unaltered PERPAR.MAC is
being edited. It is recommended that a
copy of the original PERPAR.MAC be made
and saved for future use.

If the display module is not included in the LPS support to be linked,
another background routine may be linked with BASIC, but it must be
defined in this module. See Section 8.8.1 of the BASIC/RT-11 Language
Reference Manual for instructions to define the background routine.

For the LPS routines to be accessible from the BASIC CALL statement,
the routine must be defined in a System Function Table as described in
Section 8.2 of the BASIC/RT-11 Language Reference Manual. FTBL.MAC is
a function table in source form. If any user-written assembly
language routines are also linked with BASIC and LPS software, the
routines must be defined in this function table. See Section 8.2.1 of
the BASIC/RT-11 Language Reference Manual for instructions to add the
assembly language routine definitions to the Function Table.

NOTE

Since only one .DEVICE programmed
request can be active, a special routine
SDEVHD has been added to BASIC (in the
FTBL.MAC module) to maintain a dynamic
device list. This routine is required
so that abnormal termination of BASIC
(e.g., typing CTRL/C) will return the
system to its normal state, usually by
disabling interrupts.

When writing new assembly language
routines for BASIC that would normally
use the RT-11 .DEVICE programmed
request, use the following code instead:

```
.GLOBL SDEVHD
MOV #LIST,R0
JSR PC,SDEVHD
```

If the .DEVICE list exceeds a block of
30 octal words, the error message "?DSP"
will result. The size of the block can
be changed by defining "DSPSIZ=X" in the
PERPAR.MAC file.

PERVEC.MAC is the vector definition module. It defines the hardware
addresses of the status registers and the interrupt vectors. The
standard hardware address for the LPS11 and the AR11 is 170400 (octal)

and interrupt vector is 340 (octal). The standard hardware address for the LPS11 digital I/O is LPS register+10 (octal) and interrupt vector is at LPS vector+10 (octal); for the DR11-K they are at 167770 (octal) and 300 (octal) respectively.

There are some LPS11 hardware systems with the interrupt vector at location 300 (octal). To assemble PERVEC with the interrupt vector at 300 (octal) it is necessary to delete the semicolon before the $V=0 definition in PERPAR.MAC. If the interrupt vectors are at other locations then correct the interrupt addresses by using the system editor to define $V in PERPAR equal to the interrupt address minus 300 (octal). For example, if the LPS11 interrupt vectors start at 320 (octal), define $V=20 (octal).

If the registers and interrupt vector of DR11-K are located at non-standard address, then OFFST1 and OFFST2 in PERPAR.MAC must be redefined. For example, if the register and interrupt vector addresses are at 167750 (octal) and 360 (octal), define OFFST1=-20 (octal) and OFFST2=60 (octal).

In a multiple DR11-K system there must be a difference of 10 (octal) between each unit in the interrupt and vector addresses. The status register address decreases by 10 (octal) while the vector address increases by 10 (octal). For example, in a 2 DR11-K system, when the first DR11-K is at 167750 and 360 (octal), the second one must be at 167740 and 370 (octal).

PERPAR.MAC is a parameter file, a listing follows:

```
.TITLE PERPAR -- PERIPHERAL SUPPORT PACKAGE PARAMETER MODULE.
;
; DEC-11-LBPAA-C          BASIC KERNEL V02-01
;
; COPYRIGHT (C) 1974,1975
;
; DIGITAL EQUIPMENT CORPORATION
; MAYNARD, MASSACHUSETTS 01754
;
; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO
; CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED
; AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
; DIGITAL ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT
; MAY APPEAR IN THIS DOCUMENT
;
; THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A
; LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND
; CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT
; NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY
; OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.
;
; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE
; OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
; WHICH IS NOT SUPPLIED BY DIGITAL.
;




; THE CONDITIONALS CONTAINED IN THIS MODULE AFFECT THE ASSEMBLY
; OF THE FUNCTION TABLE MODULE "FTBL.MAC".
; TO OBTAIN THE DESIRED CONDITIONAL DEFINITION(S),
; REMOVE (USING AN EDITOR) THE
; SEMI-COLON APPEARING BEFORE THE CONDITIONAL.
```

```
;
;$LSI11=0                      ;DEFINE FOR LSI-11
;
;$CAPS=0                       ;DEFINE FOR CAPS-11 SYSTEM
;
;$DISK=0                       ;DEFINE FOR RT-11
        .IFNDF $DISK
$STRNG=0                       ;DO NOT DEFINE FOR PTS BASIC WITHOUT
                               ;STRINGS,- DEFINED FOR PTS V01 WITH STRINGS
        .ENDC
        .IFDF    $DISK
$RTV2=0                        ;ASSUMES RT-11 VERSION 2 IS USED.
                               ;DO NOT DEFINE FOR RT-11 V01-15.
        .ENDC                  ; $DISK

        .IFDF    $RTV2    ;THESE ARE DISPLACEMENTS USED FOR VERSION 2
        GVECT=354        ;DISPL. TO VT11 VECTOR IN RMON
        CONFIG=300       ;DISPL. TO CONFIGURATION WORD IN RMON
;
;DSPSIZ=X
;DEFINE 'X'= TO REQUIRED SIZE OF .DEVICE LIST IN FTBL.MAC
;
        .ENDC                  ; $RTV2
;$LPS=0                        ;DEFINE FOR LPS11
;$AR11=0                       ;DEFINE FOR AR11
;
        .IFDF    $AR11
$LPS=0                         ;IMPLICIT DEFINITION
                               ;AR11 SUBSET OF LPS11
        .ENDC
;
;$DR11K=0                      ;DEFINE FOR DR11K
;
        .IFDF    $DR11K
OFFST1=0                       ;THE STANDARD DR11-K REGISTER IS AT 167770
                               ;DEFINE X, WHERE OFFST1=X, FOR NON-STANDARD
                               ;CONFIGURATION. E.G. FOR REGISTER ADDRESS AT
                               ;167750 DEFINE X=-20.
OFFST2=0                       ;THE STANDARD DR11-K VECTOR IS AT 300. DEFINE
                               ;X, WHERE OFFST2=X FOR NON-STANDARD CONFIGURATIO
                               ;E.G. FOR VECTOR AT 330, DEFINE X=30.
        .ENDC
;
;DEFINE "X" EQUAL TO TOTAL NUMBER OF DR11-K'S IN SYSTEM
;
;$NUMBR=X                      ;DEFINE FOR MULTIPLE DR11-K SYSTEM
;
        .IFNDF   $NUMBR
$NUMBR   =          1
        .ENDC
;
;
        .IFDF $LPS
;$V=0                          ;DEFINE FOR LPS WITH VECTORS STARTING
;                              ;    AT 300.  DEFAULT SETTING IS VECTORS AT
;                              ;    340. SET $V = ANY OTHER DISPLACEMENT IF
;                              ;        VECTORS START AT DISPLACEMENTS
;                              ;        OTHER THAN 0 OR 40 FROM
;                              ;        VECTOR 300

$ADC=0                         ;INCLUDE A/D ROUTINES.
$CLK=0                         ;INCLUDE CLOCK ROUTINES.
$DIO=0                         ;INCLUDE DIGITAL IO ROUTINES
$DIS=0                         ;INCLUDE DISPLAY ROUTINES.
        .ENDC                  ; $LPS
```

```
;
;$PLOT=0                        ; DEFINE FOR PLOT SUPPORT
;                               ; N O T E--$DISK MUST ALWAYS BE DEFINED FOR
;                               ;          LV11 SUPPORT.
;
        .IFDF    $PLOT
        $LV11=0                 ; INCLUDE LV11 SUPPORT
        $VT11=0                 ; INCLUDE VT11 SUPPORT
        .ENDC                   ; $PLOT
;
;$LV11=0                        ; DEFINE FOR LV11 SUPPORT
;

;
;
;$VT11=0                                      ;FOR VT11
;
;


        .IFDF    $VT11
$CLOCK=0                        ;FOR SYSTEM CLOCK (KW11L)
        .ENDC                   ; $VT11
;
;$VT55=0                        ;DEFINE FOR VT55
;


        .EOT
```

To link the LPS module with BASIC it is necessary to delete the
semicolon (;) before $AR11 (for AR11 only), $LPS (for LPS only) and
$DR11K (for DR11-K only) statements. If any of the four optional
modules are not to be included, a semicolon must be inserted before
the appropriate conditional.

Parameter	Insert semicolon before parameter if:
$ADC=0	module LPS1 or ARD1 is not to be included.
$CLK=0	module LPS2(LPS2C) or ARD2(ARD2C) is not to be included.
$DIO=0	module LPS3 or ARD3 is not to be included.
$DIS=0	module LPS4 is not to be included.

NOTE

> If DR11-K is used on a system. LPSDR
> must not be used. The total number of
> DR11-K is defined in the PERPAR.MAC
> parameter file by $NUMBR=X, where X can
> have a maximum of 16.

Using the system assembler, the sources are assembled in the following
combinations to produce the needed object modules:

Object Files	Source Files
FTBL	PERPAR,FTBL
PERVEC	PERPAR,PERVEC
RTINT	PERPAR,RTINT

After these modules have been reassembled, the LPS support can be
linked with the BASIC object modules with only the desired optional
LPS modules included in the LINK command strings.

For long programs that do not use string variables, the LPS support
may be linked with the no string object modules BASNSR, BASNSX, and
BASNSE. This no string version of BASIC with LPS support has more
free core for program array storage.

After BASLPS or BASARD has been linked, it may be loaded by the
following monitor command:

        .R BASLPS (or BASARD)

At this point the standard BASIC initial dialogue begins. See Chapter
1 of the BASIC/RT-11 Language Reference Manual for a description of
the initial dialogue.

When editing PERPAR.MAC, $DISK=0 should be enabled for BASIC/RT-11,
$LPS=0 should be enabled for BASIC with LPS11 support, $AR11=0 should
be enabled for BASIC with AR11 support, and $DR11K=0 for DR11-K
support. $ADC=0, $CLK=0, $DIO=0, and $DIS=0 should be disabled
whenever the appropriate optional LPS module (or AR11) is not to be
included. For hardware address other than the standard, make changes
as described in paragraph about PERVEC.MAC.

The procedures for building the following load modules are described
in this section:

        BASIC/RT-11   with complete LPS11 support

        BASIC/RT-11   with complete LPS11 support and LPS interrupt
                         vectors at location 300 (octal).

        BASIC/RT-11   for AR11 and DR11-K (at hardware addresses 167750
                         and 360 (octal)) support, with the ADC, DIO, CLK,
                         and DIS optional modules.

Linking instructions for both overlaying and non-overlaying versions
are given for each. Since all editing instructions assume an original
PERPAR.BAK, the edit back-up file, is renamed PERPAR.MAC to allow any
future load modules to be built from an un-edited PERPAR.MAC.

To build a load module BASLPS.SAV with complete configuration under
RT-11 including LPS support and all four optional modules, enter the
following command strings:

```
.R EDIT
*EBPERPAR. MAC$R$$
*F; $DISK=6$0AD$$
*F; $LPS=6$0AD$$
*EX$$

.R MACRO
*FTBL=PERPAR.FTBL
ERRORS DETECTED: 0
FREE CORE: 12062. WORDS

*PERVEC=PERPAR.PERVEC
ERRORS DETECTED: 0
FREE CORE: 12439. WORDS
```

```
*RTINT=PERPAR,RTINT
ERRORS DETECTED: 0
FREE CORE: 12568 WORDS

*^C

.R PIP
*PERPAR.MAC=PERPAR.BAK/R
*^C
```

The following instructions will create a BASLPS.SAV with overlaying
which has the maximum usable area.

```
.R LINK
*BASLPS,BASLPS=BASICR,FPMP,FTBL,PERVEC,RTINT/B:400/T/C
TRANSFER ADDRESS =
GO
*LPS0,LPS1,LPS2,LPS3,LPS4/C
*BASICF/O:1/C
*BASICX/O:1/C
*BASICH/O:2


*
```

To link a non-overlaying BASLPS.SAV which will have   increased
execution speed the following commands should be given to link:

```
.R LINK
*BASLPS,BASLPS=BASICR,FPMP,BASICE,BASICX/B:400/C
*FTBL,PERVEC,RTINT/C
*LPS0,LPS1,LPS2,LPS3,LPS4,BASICH


*
```

Complete configuration for building a load module with interrupt
vector at location 300 (octal) - These instructions are the same as
the preceding instructions except that a $V=0 parameter definition in
PERPAR.MAC will be enabled.

```
.R EDIT
*EBPPERPAR.MAC$R$$
*F; $DISK=0$0AD$$
*F; $LPS=0$0AD$$
*F; $V=0$0AD$$
*FX$$

.R MACRO
*FTBL=PERPAR,FTBL
ERRORS DETECTED: 0
FREE CORE: 12058 WORDS

*PERVEC=PERPAR,PERVEC
ERRORS DETECTED: 0
FREE CORE: 12439 WORDS

*RTINT=PERPAR,RTINT
ERRORS DETECTED: 0
FREE CORE: 12564 WORDS

*^C
```

```
.R PIP
*PERPAR.MAC=PERPAR.BAK/R
*^C
```

The following procedure can create the overlaying version.

```
.R LINK
*BASLPS,BASLPS=BASICR,FPMP,FTBL,PERVEC,RTINT/B:400/T/C
TRANSFER ADDRESS =
GO
*LPS0,LPS1,LPS2,LPS3,LPS4/C
*BASICE/O:1/C
*BASICX/O:1/C
*BASICH/O:2


*
```

To create the non-overlaying version the following link commands
should be given:

```
.R LINK
*BASLPS,BASLPS=BASICR,FPMP,BASICE,BASICX/B:400/C
*FTBL,PERVEC,RTINT/C
*LPS0,LPS1,LPS2,LPS3,LPS4,BASICH


*
```

Partial Configuration - To build a load module BASARD.SAV under RT-11
for AR11 and DR11-K support (at hardware address 167750 and 360
(octal) which includes the ADC, DIO, CLK, and DIS routines, enter the
following command strings:

```
.R EDIT
*EBPERPAR.MAC$R$$
*F;$DISK=0$0AD$$
*F;$AR11=0$0AD$$
*F;$DR11K=0$0AD$$
*EX$$

.R EDIT
*EBPERVEC.MAC$R$$
*FOFFST1=0$-C-20$$
*FOFFST2=0$-C60$$
*EX$$

.R MACRO
*FTBL=PERPAR,FTBL
ERRORS DETECTED: 0
FREE CORE: 12054. WORDS

*PERVEC=PERPAR,PERVEC
ERRORS DETECTED: 0
FREE CORE: 12423. WORDS

*RTINT=PERPAR,RTINT
ERRORS DETECTED: 0
FREE CORE: 12560. WORDS

*^C
```

```
.R PIP
*PERPAR.MAC=PERPAR.BAK/R
*PERVEC.MAC=PERVEC.BAK/R
*^C

.R LINK
*BASARD,BASARD=BASICR,FPMP,FTBL,PERVEC,RTINT/B:400/T/C
TRANSFER ADDRESS =
GO
*LPS0,ARD1,ARD2,ARD3,LPS4/C
*BASICE/O:1/C
*BASICX/O:1/C
*BASICH/O:2


*
```

This procedure will create an overlaying version of BASLPS.SAV. The
following command strings may be used to link a non-overlaid version
of BASIC with equivalent LPS support:

```
.R LINK
*BASARD,BASARD=BASICR,FPMP,BASICE,BASICX/B:400/C
*FTBL,PERVEC,RTINT/C
*LPS0,ARD1,ARD2,ARD3,LPS4,BASICH


*
```

The Laboratory Peripheral System support may also be purchased in
source form.   The following nine source files are provided in the
BASIC Extensions package.

```
        LPS0.MAC
        LPS1.MAC
        LPS2.MAC
        LPS3.MAC
        LPS4.MAC
        PTBL.MAC
        PERVEC.MAC
        RTINIT.MAC
        PERPAR.MAC
```

The following table lists the assembly parameters for each module:

Source File	Conditionals	Define for Systems with:
LPS0.MAC	None	
LPS1.MAC	$AR11	AR11 hardware
LPS2.MAC	CYC50	50 Hz line frequency (60 Hz is default)
	$AR11	AR11 hardware
LPS3.MAC	$AR11	AR11 hardware
LPS4.MAC	None	

Source File	Conditionals	Define for Systems with:
FTBL.MAC	$ADC	LPS1 (ARD1 for AR11)
	$CLK	LPS2 (ARD2 for AR11)
	$DIO	LPS3 (ARD3 for AR11)
	$DIS	LPS4
	$LPS	$LPS0 (all systems with LPS support)
	$AR11	AR11 support
	$VT11	VT11 support
	$DISK	RT-11
	$VT55	VT55 support
PERVEC.MAC	$LPS	LPS11 hardware
	$V	LPS (or AR11) interrupts not at location 340 (octal)
	$VT11	VT11 support
	$AR11	AR11 hardware
	$DR11K	DR11-K hardware
	$NUMBR=X	Multiple (X) DR11-K hardware
	OFFST1	First DR11-K interrupt address not at 167770 (octal)
	OFFST2	First DR11-K vector address not at 300 (octal)
RTINT.MAC	$DIS	LPS4

The conditional $RTV2 is present in all modules to force subtitles in assembly listings and enable RT-11 V02 system macro processing.

To assemble the LPS from the sources all the LPS files should be transferred to the system device using PIP, and then the following command should be given to the RT-11 MACRO assembler; if the line current is 50 Hz, the following commands should be used before calling MACRO:

```
.R EDIT
*EWPARAM MAC$$
*ICYC50=0
$$
*EX$$

.R MACRO
*LPS0=PERPAR,LPS0            (for LPS11 only - define $LPS in
ERRORS DETECTED: 0           PERPAR.MAC -)
FREE CORE: 12144. WORDS

*LPS1=PERPAR,LPS1
ERRORS DETECTED: 0
FREE CORE: 11980. WORDS

*LPS2=PERPAR,LPS2            (for 60 Hz clock)
ERRORS DETECTED: 0
FREE CORE: 12152. WORDS

*LPS2C=PERPAR,PARAM,LPS2     (for 50 Hz clock)
ERRORS DETECTED: 0
FREE CORE: 12148. WORDS

*LPS3=PERPAR,LPS3
ERRORS DETECTED: 0
FREE CORE: 12072. WORDS
```

```
*LPS4=PERPAR,LPS4
ERRORS DETECTED: 0
FREE CORE: 12004. WORDS

*ARD1=PERPAR,LPS1                    (for   AR11&   DR11-K   only-define
ERRORS DETECTED: 0                   $AR11& $DR11K in PERPAR.MAC)
FREE CORE: 11980. WORDS

*ARD2=PERPAR,LPS2                    (for 60 Hz clock)
ERRORS DETECTED: 0
FREE CORE: 12152. WORDS

*ARD2C=PERPAR,PARAM,LPS2             (for 50 Hz clock)
ERRORS DETECTED: 0
FREE CORE: 12148. WORDS

*ARD3=PERPAR,LPS3
ERRORS DETECTED: 0
FREE CORE: 12072. WORDS

*
```

Following is a listing of PERVEC.MAC which contains the interrupt
vector location for the LPS and GT44 hardware:

```
.TITLE PERVEC VECTOR DEFINITION MODULE FOR BASIC SUPPORT PACKAGES.
;
; DEC-11-LBPVA-B          BASIC KERNEL V02-01
;
; COPYRIGHT (C) 1974,1975
;
; DIGITAL EQUIPMENT CORPORATION
; MAYNARD, MASSACHUSETTS 01754
;
; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO
; CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED
; AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
; DEC ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT
; MAY APPEAR IN THIS DOCUMENT.
;
; THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A
; LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND
; CAN BE COPIED (WITH INCLUSION OF DEC'S COPYRIGHT
; NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY
; OTHERWISE BE PROVIDED IN WRITING BY DEC.
;
; DEC ASSUMES NO RESPONSIBILITY FOR THE USE
; OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
; WHICH IS NOT SUPPLIED BY DEC.




; THIS MODULE DEFINES THE HARDWARE ADRESSES USED BY
; SUCH HARDWARE AS THE "LPS", "AR11", "DR11-K",
; THE "VT11"(GT40) AND THE "LV11".
; IF THE VECTORS FOR THESE DEVICES SHOULD CHANGE
; THIS MODULE MUST BE EDITED TO REFLECT THE CHANGE.
;
;       DEFINE   $LPS    FOR LPS/AR11 SUPPORT IN PERPAR
;                $AR11   FOR AR11 MODS.
;
```

```
;IF SYSTEM HAS DR11-K, $NUMBR MUST BE DEFINED EQUAL TO TOTAL NUMBER
;OF DR11-K'S IN SYSTEM.
;                  DEFINE $DR11K & $NUMBR  FOR DR11-K
;
; *********     NOTE    *********
; IF DR11-K(S) IS (ARE) PRESENT IN SYSTEM ALONG WITH THE LPS11
; THE LPSDR CANNOT BE USED.
; *******************************
;
;


        .IFDF  $LPS
        .IFNDF  $V
$V=40
        .ENDC
        .GLOBL LPSAD,LPSADB,LPSDR,LPSDMA
        .GLOBL LPSCKS,LPSPB,LPSDRS,LPSDIB
        .GLOBL LPSDOR,DRSTOF
        .GLOBL LPDISS,LPDISX,LPDISY
        .GLOBL CKL1VA,CKL1P,DRSIVA,DRSIP,LPSIVA,LPSIP


; DEVICE EQUATES:

LPSAD   =       170400   ;LPS A/D STATUS REG.
LPSADB  =       170402   ;LPS A/D BUFFER LED REG.
LPSCKS  =       170404   ;LPS CLOCK STATUS REG.
LPSPB   =       170406   ;LPS CLOCK BUFFER PRESET REG.
        .IFNDF  $AR11
LPSDR   =       170410   ;LPS DIGITAL I/O STATUS REG.
LPSDRS  =       LPSDR
LPSDIB  =       170412   ;LPS DIGITAL INPUT REG.
LPSDOR  =       170414   ;LPS DIGITAL OUTPUT REG.
LPDISS  =       170416   ;LPS DISPLAY STATUS REG.
LPDISX  =       170420   ;LPS DISPLAY REG. X
LPDISY  =       170422   ;LPS DISPLAY REG. Y
        .ENDC
        .IFDF  $AR11
LPDISS  =       170410
LPDISX  =       170412
LPDISY  =       170414
        .ENDC
LPSDMA  =       170436   ;LPS DMA REGG.
;
; DEFINE "OFFST1" IF DR11-K NOT AT STANDARD LOCATION. ALL DR11-K'S
; MUST BE NEXT TO EACH OTHER(REGISTER ADDRESS DIFFERENCE=-10 ONLY)
;
        .IFDF  $DR11K
OFFST1=0
LPSDR   =       167770+OFFST1
LPSDRS  =       LPSDR
LPSDIB  =       167772+OFFST1
LPSDOR  =       167774+OFFST1
        .ENDC
        .IFG   $NUMBR-1
OFFST1=0
LPSDR   =       167770+OFFST1
LPSDRS  =       LPSDR
LPSDIB  =       167772+OFFST1
LPSDOR  =       167774+OFFST1
        .ENDC
```

```
; INTERRUPT VECTOR PAIRS:


CKLIVA  =       304+$V  ;ADR. OF CLOCK INTERRUPT VECTOR
CKLIP   =       306+$V  ;ADR. OF CLOCK INT.      PRIORITY
;
        .IFNDF  $AR11

DRSIVA  =       310+$V  ;ADR. OF DRS INPUT INT. VECTOR
DRSIP   =       312+$V  ;ADR. OF DRS INPUT INT. PRIORITY.
        .ENDC
;
; DEFINE "OFFST2" IF DR11-K NOT AT STANDARD VECTOR LOCATION.
;
        .IFDF   $DR11K
OFFST2=0
DRSIVA  =       300+OFFST2
DRSIP   =       302+OFFST2
        .ENDC
;

LPSIVA  =       300+$V  ;ADR. OF THE A/D INT. VECTOR.
LPSIP   =       302+$V  ;ADR. OF THE INT.PRIORITY.

        .ENDC   ; $LPS

        .IFDF   $TABLT              ;PROTOTYPE WRITING TABLET
        .GLOBL  TSR,TINT,TVEC,XLOC,YLOC
TSR     =       164040              ;WRITING TABLET CSR
XLOC    =       TSR+2               ;X LOCATION OF TURTLE
YLOC    =       XLOC+2              ;Y LOCATION OF TURTLE
TVEC    =       270                 ;INTERRUPT VECTOR
        .ENDC
        .IFDF   $VT11   ;GT40
        .GLOBL  DPC,DSR,DISX,DISY,GTVECT

DPC     =       172000  ;VT11 DISPLAY PC
DSR     =       DPC+2   ;VT11 DISPLAY STATUS REG
DISX    =       DSR+2   ;VT11 X STATUS REG
DISY    =       DISX+2  ;VT11 Y STATUS REG
GTVECT  =       320     ;ADR. OF VT11 [GT40 (GT44)] INTERRUPT
                        ;VECTOR LIST. REDEFINING GTVECT
                        ;REDEFINES THE ENTIRE SET
                        ;OF DISPLAY PROCESSOR INT. VECTORS.
;GTVECT:                ;DISPLAY STOP VECTOR
;GTVECT+4:              ;LIGHT PEN HIT VECTOR
;GTVECT+10:             ;DISPLAY TIME OUT VECTOR
        .ENDC   ; $VT11

        .IFDF   $LV11   ; LV11 SUPPORT
        .GLOBL  LVCS,LVDB

LVCS    =       177514  ; LV11 STATUS REGISTER
LVDB    =       LVCS+2  ; LV11 DATA BUFFER
        .ENDC
;
;
;THE FOLLOWING ARE GLOBAL CONSTANTS WHICH ARE REDEFINED BASED ON
;THE SYSTEM CONFIGURATION(LIKE LPS/AR11 ETC.).
;
;
        .GLOBL  $LPS11,$LPS12,$LPS13     ;FOR LPS1 MODULE
```

```
$LPS11:
        .IFNDF  $AR11               ;LPS
        .WORD   77,4                ;77=63.=CHAN, 4=IGAIN
$LPS13=6000                         ;FOR AUTO-GAIN RANGE CHECKING
        .ENDC
        .IFDF   $AR11
        .WORD   17,2
$LPS13=1000
        .ENDC
;
$LPS12:
        .IFNDF  $AR11               ;LPS
        .WORD   77,100              ;77=63.=CHAN, 100=64.=NSC
        .WORD   177777,17,4         ;177777=65535.=NPTS, 17=15.=MODE, 4=IGAIN
        .ENDC
        .IFDF   $AR11                        ;AR11
        .WORD   17,20               ;17=15.=CHAN, 20=16.=NSC
        .WORD   177777,2,2          ;177777=65535.=NPTS, 2=MODE, 2=IGAIN
        .ENDC
;
        .GLOBL  $LPS21              ;FOR LPS2 MODULE
$LPS21:
        .IFNDF  $AR11               ;LPS
        .WORD   7,17,177777         ;7=RATE, 17=15.=MODE, 177777=65535.=PRESET
        .ENDC
        .IFDF   $AR11                        ;AR11
        .WORD   7,11,377            ;7=RATE, 11=9.=MODE, 377=255.=PRESET
        .ENDC
;
        .GLOBL  $LPS31    ;FOR LPS3 MODULE
$LPS31:
        .WORD   177777              ;177777=65535.=N
        .WORD   177777              ;177777=65535.=MASK
        .WORD   $NUMBR-1            ;NUMBER OF DR11-K'S IN SYSTEM. DEFAULT=1
;
        .GLOBL  $LPS32
$LPS32: 177777                      ;177777=65535.=SW
        177777                      ;177777=65535.=VAR(WRITE)
;
        .GLOBL  $LPS41,$LPS42,$LPS43,$LPS44        ;FOR LPS4 MODULE
        .IFNDF  $AR11               ;LPS
$LPS41: .WORD   10000               ;10000=4096.=XMAX/YMAX
$LPS42: .WORD   7777                ;7777=4095.=N
$LPS43: .WORD   7777                ;7777=4095.=I
$LPS44: .WORD   177777              ;177777=65535.=NP
        .ENDC
        .IFDF   $AR11                        ;AR11
$LPS41: .WORD   2000                ;2000=1024.=XMAX/YMAX
$LPS42: .WORD   1777                ;1777=1023.=N
$LPS43: .WORD   1777                ;1777=1023.=I
$LPS44: .WORD   177777              ;177777=65535.=NP
        .ENDC
;
;
        .GLOBL  DRSON,DRSTOF                ;DIGITAL I/O STORAGE VARIES WITH
                                    ;TOTAL NUMBER OF DR11-K'S IN SYSTEM
DRSON:
        .WORD   0                   ;DRSON:0:      DRS OPERATION IN PROGRESS
        .WORD   0                   ;BCDON:2:      BCD/BINARY SWITCH
        .WORD   0                   ;DRSBUF:4:     DRS BUFFER DESCRIPTOR ADD.
        .WORD   0                   ;DRSNPT:6:     DRS NUMBER OF POINTS
        .WORD   0                   ;MASK:10:      DRS MASK FOR INPUT
        .WORD   0                   ;DRTBUF:12:    DRS TIME BUFFER DESC.
```

```
DRSTOF=.-DRSON
;
            .IFG        $NUMBR-1
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-2
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-3
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-4
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-5
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-6
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-7
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
            .WORD       0                   ;MASK:10:        DRS MASK FOR INPUT
            .WORD       0                   ;DRTBUF:12:      DRS TIME BUFFER DESC.
            .ENDC
            .IFG        $NUMBR-10
            .WORD       0                   ;DRSON:0:        DRS OPERATION IN PROGRESS
            .WORD       0                   ;BCDON:2:        BCD/BINARY SWITCH
            .WORD       0                   ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
            .WORD       0                   ;DRSNPT:6:       DRS NUMBER OF POINTS
```

```
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
        .IFG     $NUMBR-11
        .WORD    0                    ;DRSON:0:        DRS OPERATION IN PROGRESS
        .WORD    0                    ;BCDON:2:        BCD/BINARY SWITCH
        .WORD    0                    ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
        .WORD    0                    ;DRSNPT:6:       DRS NUMBER OF POINTS
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
        .IFG     $NUMBR-12
        .WORD    0                    ;DRSON:0:        DRS OPERATION IN PROGRESS
        .WORD    0                    ;BCDON:2:        BCD/BINARY SWITCH
        .WORD    0                    ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
        .WORD    0                    ;DRSNPT:6:       DRS NUMBER OF POINTS
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
        .IFG     $NUMBR-13
        .WORD    0                    ;DRSON:0:        DRS OPERATION IN PROGRESS
        .WORD    0                    ;BCDON:2:        BCD/BINARY SWITCH
        .WORD    0                    ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
        .WORD    0                    ;DRSNPT:6:       DRS NUMBER OF POINTS
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
        .IFG     $NUMBR-14
        .WORD    0                    ;DRSON:0:        DRS OPERATION IN PROGRESS
        .WORD    0                    ;BCDON:2:        BCD/BINARY SWITCH
        .WORD    0                    ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
        .WORD    0                    ;DRSNPT:6:       DRS NUMBER OF POINTS
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
        .IFG     $NUMBR-15
        .WORD    0                    ;DRSON:0:        DRS OPERATION IN PROGRESS
        .WORD    0                    ;BCDON:2:        BCD/BINARY SWITCH
        .WORD    0                    ;DRSBUF:4:       DRS BUFFER DESCRIPTOR ADD.
        .WORD    0                    ;DRSNPT:6:       DRS NUMBER OF POINTS
        .WORD    0                    ;MASK:10:        DRS MASK FOR INPUT
        .WORD    0                    ;DRTBUF:12:      DRS TIME BUFFER DESC.
        .ENDC
;
.END
```

B.2   BASIC/CAPS-11

B.2.1   LPS Support

The Laboratory Peripheral System (LPS) support for BASIC is supplied
in ten binary files.

LPS0.OBJ	Required	LPS kernel module for LPS11 or AR11 and DR11-K.
LPS1.OBJ	Optional	Analog to digital conversion for LPS11.
ARD1.OBJ	Optional	Analog to digital conversion for AR11 and DR11-K.

LPS2.OBJ ⎫		Real-time clock (60 Hz line frequency) for LPS11.
ARD2.OBJ ⎪	One is required	Real-time clock (60 Hz line frequency) for AR11 and DR11-K.
LPS2C.OBJ ⎬		Real-time clock (50 Hz line frequency) for LPS11.
ARD2C.OBJ ⎭		Real-time clock (50 Hz line frequency) for AR11 and DR11-K.
LPS3.OBJ	Optional	Digital input/output for LPS11.
ARD3.OBJ	Optional	Digital input/output for AR11 and DR11-K.
LPS4.OBJ	Optional	Display for LPS11 or AR11 and DR11-K.

The standard BASIC/CAPS binary kit contains all the object modules required to link a version of BASIC/CAPS that contains LPS support (with all four optional LPS modules).

There are also the following files which are provided in source form in all kits:

FTBL.PAL	Function Table Module
PERVEC.PAL	Vector Definition Module
BASINT.PAL	Interface Module
PERPAR.PAL	Parameter file

NOTE

BASIC with LPS support requires a PDP-11 with 16K or more of memory. The procedures in this section assume the user has at least 16K of memory and has reconfigured his CAPS-11 system, along with PAL and LINK, for 16K.

To create a version of BASIC/CAPS with complete LPS support, no GT support, and a standard hardware configuration, it is only necessary to link the supplied object modules.

To create a customized version of BASIC/CAPS with LPS support, the parameter file PERPAR.PAL is edited and assembled with FTBL.PAL, PERVEC.PAL and BASINT. The three object modules produced are then linked with the LPS and BASIC object modules to produce a load module.

NOTE

All of the procedures in this section assume that an unaltered PERPAR.PAL is being edited. It is recommended that a copy of the original PERPAR.PAL be made and saved for future use.

The BASINT.PAL interface module should be used with all versions of BASIC/CAPS. If the display module is not included in the LPS support to be linked, another background routine may be linked with BASIC but it must be defined in this module.

For the LPS routines to be accessible from the BASIC CALL statement, the routine must be defined in a System Function Table. FTBL.PAL is a function table in source form. If any user-written assembly language routines are also linked with BASIC the routines must be defined in this function table.

PERVEC.PAL is the vector definition module. It defines the hardware addresses of the status registers and the interrupt vectors. The standard hardware address for the LPS (or AR11) interrupt vector is 340 (octal). In PDP-11E10 machines with LPS (or AR11) support, however, the interrupt vector is location 300 (octal). To assemble PERVEC with the interrupt vector at 300 (octal) it is necessary to delete the semicolon before the $V=0 definition in PERPAR.PAL. If the interrupt locations are at another location in memory, correct the interrupt addresses by using the system editor to define $V in PERPAR equal to the interrupt address minus 300 (octal). For example, if the LPS (or AR11) interrupt vectors start at 320 (octal) define $V=20 (octal). A listing of PERVEC.PAL is printed at the end of section B.1.2. To link the LPS (or AR11) module with BASIC it is necessary to delete the semicolon (;) before the $LPS=0 (or $AR11=0) statement in PERPAR.PAL. If any of the four optional modules are not to be included, a semicolon (;) must be inserted before the appropriate conditional.

Parameter	Insert ; before parameter if
$ADC=0	module LPS1 or ARD1 is not to be included.
$CLK=0	module LPS2 (LPS2C) or ARD2 (ARD2C) is not to be included.
$DIO=0	module LPS3 or ARD3 is not to be included.
$DIS=0	module LPS4 is not to be included.

Using the system assembler PAL, the sources are assembled in the following combinations to produce the needed LPS object modules:

Object File	Source Files
FTBL	PERPAR, FTBL
PERVEC	PERPAR, PERVEC
BASINT	PERPAR, BASINT

After these modules have been assembled, the LPS support may be linked with the BASIC object modules with only the desired optional LPS modules included in the LINK command strings.

After BASLPS has been linked it may be loaded by the following monitor command:

        .R BASLPS

At this point the standard BASIC initial dialogue occurs.

As part of the initial dialogue, BASIC prints:

    USER FNS LOADED

This message occurs whenever BASIC has been linked with LPS support.

When editing PERPAR.PAL, $LPS=0 should be enabled for BASIC with any LPS support (or $AR11 for AR11 support), and $ADC=0, $CLK=0, $DIO=0, and $DIS=0 should be disabled whenever the appropriate optional LPS module is not to be included. In addition, $V=0 should be enabled for any PDP-11 with LPS (or AR11) hardware interrupt located at 300 (octal) instead of 340 (octal). Most PDP-11E10 with LPS (or AR11) require the defining of the $V=0 assembly parameter. For hardware addresses other than 300 or 340, define $V as described in paragraph about PERVEC.PAL. Linking BASIC/CAPS with LPS Support

The procedures for building the following load modules are described below:

    BASIC/CAPS   with complete LPS support.

    BASIC/CAPS   with complete LPS support and LPS interrupt
                     vectors at location 300 (octal).

    BASIC/CAPS   with only the ADC and DIS optional display
                     modules.

Since all editing instructions assume an original PERPAR.PAL, a copy of the original file should be preserved to allow any future load modules to be built from an unedited PERPAR.PAL.

In all the following examples the user can substitute the module LPS2C for the module LPS2 if the line frequency is 50 Hz instead of 60 Hz.

Complete Configuration

To build a load module BASLPS.SLO under CAPS-11 including LPS support and all four optional modules, enter the following command strings:

            (MOUNT A SCRATCH CASSETTE ON DRIVE 1)

    .Z 1:
            (TYPE <CR>)
    .Z 1:
            (MOUNT A SECCOND SCRATCH CASSETTE ON DRIVE 1)
    .R EDIT
    *EW1:PERPAR.LPS$$
            (WHEN CASSETTE ON DRIVE 0 REWINDS, MOUNT
            CASSETEE CONTAINING PERPAR.PAL ON DRIVE 0)
    *ER0:PERPAR.PAL$$R$$
    *F:$CAPS=0$$0AI$$
    *F:$LPS=0$$0AI$$
    *EX$$

    .R PAL16

    *0:FTBL.OBJ/P=1:PERPAR.LPS/P,FTBL.PAL/P

    1?

    1?

```
PASS 2

0?

1?

1?

000000 ERRORS


*0:PERVEC.OBJ=1:PERPAR.LPS/P,PERVEC.PAL/P

1?

1?

PASS 2

1?

1?

000000 ERRORS

1?

1?

PASS 2

1?

1?

000000 ERRORS


*^C             (Mount CAPS-11 system cassette on unit 0)
                (Mount scratch cassette on unit 1)

                (Mount BASIC object module cassette containing
                BASICR.OBJ
                on unit 1)

.R LINK
                (When unit 0 has rewound, mount zeroed scratch
                cassette
                on unit 0)

*1:BASLPS.SLO,LP:=0:BASICR,FPMP,BASICE,BASICX,BASICS/B:400/C
,FTBL/P,PERVEC,BASINT,LPS0/P,LPS1,LPS2,LPS3,LPS4,BASICH/P

0?             (Mount cassette with FTBL.OBJ on unit 0)

0?             (Mount cassette with LPS0.OBJ on unit 0)

0?             (LOAD MAP PRINTED)

PASS 2

0?             (Mount cassette with FTBL.OBJ on unit 0)
```

```
O?                  (Mount cassette with LPS0.OBJ on unit 0)

O?                  (Mount cassette with BASICH.OBJ on unit 0)

*^C                 (Done. Cassette on unit 0 contains a new
                    BASLPS.SLO
                    which is BASIC/CAP^C plus all LPS support-interrupt
                    vectors
                    at 340 (octal)).
```

Complete Configuration
Interrupt vectors at location 300 (octal)

These instructions are the same as the preceding instructions
except that a $V=0 parameter definition in PERPAR.PAL is
enabled.
To change the interrupt vector location, the file PERPAR.PAL
must be edited to enable the $V=0 parameter definition. The
new PERPAR is then used to reassemble PERVEC.PAL to
redefine the vector locations.

```
                  Ⓢ represents the ALTMODE key
                  (Mount CAPS-11 system cassette on unit 0)
                  (Mount scratch cassette on unit 1)
   Z 1:
                  (Mount a second scratch cassette on unit 1)
 .Z 1:

 .R EDIT
*EW1:PERPAR.LPSⓈⓈ
                  (Mount BASIC object cassette containing PERPAR.PAL
                  on unit 0)
*ER0:PERPAR.PALⓈⓇⓈⓈ
*F;$CAPS=0ⓈOAIⓈⓈ
*F;$LPS=0ⓈOAIⓈⓈ
*F;$V=0ⓈOAIⓈⓈ
*EXⓈⓈ
                  (Mount CAPS-11 system cassette on unit 0)

 .R PAL16

*FTBL/P=1:PERPAR.LPS/P,FTBL.PAL/P

 1?               (Type any keyboard character)

 1?               (Mount cassette with FTBL.PAL on unit 1)

PASS 2

 0?               (Mount second scratch cassette on unit 0)

 1?               (Mount cassette with new PERPAR.PAL on unit 1)

 1?               (Mount cassette with FTBL.PAL on unit 1)

000000 ERRORS


*PERVEC=1:PERPAR.LPS/P,PERVEC.PAL/P

 1?               (Mount cassette with PERPAR.LPS on unit 1)

 1?               (Mount cassette with PERVEC.PAL on unit 1)
```

PASS 2

1?   (Mount cassette with PERPAR.LPS on unit 1)

1?   (Mount cassette with PERVEC.PAL on unit 1)
                  (Now the cassette on unit 0 has a new PERVEC.OBJ)
000000 ERRORS

*BASINT=1:PERPAR.LPS/P,BASINT.PAL/P

1?   (Mount cassette with PERPAR.LPS on unit 1)

1?   (Mount cassette with BASINT.PAL on unit 1)
                  (Cassette on unit 0 now contains a new BASINT.OBJ)
PASS 2

1?   (Mount cassette with PERPAR.LPS on unit 1)

1?   (Cassette on unit 0 now contains a new BASINT.OBJ)

000000 ERRORS

*^C
                  (Mount CAPS-11 system cassette on unit 0)
                  (Mount cassette containing BASICR.OBJ on unit 1)

 .R LINK
                  (When unit 0 has rewound, mount cassette with new
                  PERPAR.PAL on unit 1)

                  (Mount cassette with BASICR.OBJ on unit 0)

*1:BASLPS.SLO,LP:=BASICR,FPMP,BASICE,BASICX,BASICS/B:400/C
,FTBL/P,PERVEC,BASINT,LPS0/P,LPS1,LPS2,LPS3,LPS4,BASICH/P

0?   (Mount cassette with new FTBL.OBJ on unit 0)

0?   (Mount cassette with LPS0.OBJ on unit 0)

0?   (Mount cassette with BASICH.OBJ on unit 0)

                  (LOAD MAP PRINTED)
PASS 2

0?   (Mount cassette with new FTBL.OBJ on unit 0)

0?   (Mount cassette with LPS0.OBJ on unit 0)

0?   (Mount cassette with BASICH.OBJ on unit 0)

*^C   (Done.   New version of BASLPS.SLO   with   LPS
                  interrupt vectors at 300 is on cassette 1)

Partial Configuration

To build a load module BASLPS.SLO of BASIC/CAPS which includes only
the ADC and display routines, enter the following command strings:

                  (Mount CAPS-11 system cassette on unit 0)
                  (Mount scratch cassette on unit 1)

```
.Z 1:
                    (Mount second scratch cassette on unit 1)
.Z 1:

.R EDIT
*EW1:PERPAR.LPS@@
*ERO:PERPAR.PAL@@@@
*ERO:PERPAR.PAL@@@@
*F;$CAPS=0@0AI@@
*F;$LPS=0@0AI@@
*F$CLK=0@0AI;@@
*F$DIO=0@0AI;@@
*EX@@
                    (Mount CAPS-11 system cassette on unit 0)
```

For AR11 support, enter the following command strings after mounting
second scratch cassette on unit 1:

```
.R PAL16

*FTBL/P=1:PERPAR.LPS/P,FTBL.PAL/P

    1?              (Type any keyboard character)

    1?              (Mount cassette with FTBL.PAL on unit 1)

PASS 2

    0?              (Mount second scratch cassette on unit 0)

    1?              (Mount cassette with new PERPAR.PAL on unit 1)

    1?              (Mount cassette with FTBL.PAL on unit 1)

000000 ERRORS
                    (Cassette on unit 0 now contains a new FTBLL.OBJ)

*PERVEC=1:PERPAR.LPS/P,PERVEC.PAL/P

    1?              (Mount cassette with PERPAR.LPS on unit 1)

    1?              (Mount cassette with PERVEC.PAL on unit 1)

PASS 2

    1?              (Mount cassette with PERPAR.LPS on unit 1)

    1?              (Mount casset with PERVEC.PAL on unit 1)
                    (Now the cassette on unit 0 has a new PERVEC.OBJ)
000000 ERRORS


*BASINT=1:PERPAR.LPS/P,BASINT.PAL/P

    1?              (Mount cassette with PERPAR.LPS on unit 1)

    1?              (Mount cassette with BASINT.PAL on unit 1)
                    (Cassette on unit 0 now contains a new BASINT.OBJ)

PASS 2

    1?

    1?
```

        000000 ERRORS

        *^C
                                (Mount CAPS-11 system cassette on unit 0)   (Mount
                                cassette containing BASICR.OBJ on unit 1)
        .R LINK

                                (When   unit   0   has   rewound,   mount   cassette
                                containing new PERPAR.PAL on unit 0)

        *1:BASLPS.SLO,LP:=BASICR,FPMP,BASICE,BASICX,BASICS/B:400/C
        ,FTBL/F,PERVEC,BASINT,LPS0/F,LPS1,LPS4,BASICH/F

        0?   (Mount cassette with new FTBL.OBJ on unit 0)

        0?   (Mount cassette with LPS0.OBJ on unit 0)

        0?   (Mount cassette with BASICH.OBJ on unit 0)

        PASS 2   (LOAD MAP PRINTED)

        0?   (Mount cassette with new FTBL.OBJ on unit 0)

        0?   (Mount cassette with LPS0.OBJ on unit 0)

        0?   (Mount cassette with BASICH.OBJ on unit 0)

        *^C   (Done. New version of BASLPS.SLO with only ADC
                                and display routines, interrupt vectors at 340, is
        .   on cassette 1)

Assembling LPS, AR11 Support from the Sources

The Laboratory Peripheral System support may also be purchased in
source   form.   The   following   nine   source files are provided. (The
source files for FTBL, BASINT, PERPAR, and PERVEC are provided with
the binary kit.)

        LPS0.PAL
        LPS1.PAL
        LPS2.PAL
        LPS3.PAL
        LPS4.PAL
        FTBL.PAL
        PERVEC.PAL
        BASINT.PAL
        PERPAR.PAL
The following table lists the assembly parameters for each module:

Source File	Conditionals	Define for Systems with:
LPS0.PAL	None	
LPS1.PAL	$AR11	AR11 hardware
	$CAPS	PTS hardware
LPS2.PAL	CYC50	50 Hz line frequency (60 Hz is default)
	$AR11	AR11 hardware
	$CAPS	PTS hardware
LPS3.PAL	$AR11	AR11 hardware
LPS4.PAL	None	

Source File	Conditionals	Define for Systems with:
FTBL.PAL	$ADC	LPS1 (ARD1 for AR11)
	$CLK	LPS2 (ARD2 for AR11)
	$DIO	LPS3 (ARD3 for AR11)
	$DIS	LPS4
	$LPS	$LPS0 (all systems with LPS support)
	$AR11	AR11 support
	$VT11	VT11 support
	$DISK	RT-11
	$VT55	VT55 support
PERVEC.PAL	$LPS	LPS11 hardware
	$V	LPS interrupts not at location 340 (octal)
	$VT11	VT11 support
	$AR11	AR11 hardware
	$DR11K	DR11-K hardware
	$NUMBR=X	Multiple (X) DR11-K hardware
	OFFST1	First DR11-K interrupt address not at 167770 (octal)
	OFFST2	First DR11-K vector address not at 300 (octal)
BASINT.PAL	$DIS	LPS4

To assemble the LPS object modules from the sources, use the following command strings:

```
.R EDIT
*EW1:PARAM.PAL⊗⊛
*ICYC50=2
⊗⊛
*EX⊗⊛


.R PAL16

*1:LPS0.OBJ=0:PERPAR.LPS/F,LPS0.PAL/F
```

O?   (Mount cassette with PERPAR.PAL on unit 0)

O?   (Mount cassette with LPS0.PAL on unit 0)

```
PASS 2
```

O?   (Mount cassette with PERPAR.PAL on unit 0)

O?   (Mount cassette with LPS0.PAL on unit 0)

```
000000 ERRORS

*1:LPS1.OBJ=0:PERPAR.LPS/F,LPS1.PAL/F
```

O?   (Mount cassette with PERPAR.PAL on unit 0)

O?   (Mount cassette with LPS1.PAL on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS1.PAL
                                      on unit 0)

000000 ERRORS

*1:LPS2.OBJ=0:PERPAR.LPS/P,LPS2.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

000000 ERRORS

For line frequency of 50 Hz, use LPS2C instead of LPS2.

*1:LPS2C.OBJ=0:PERPAR.LPS/P,PARAM.PAL/P,LPS2.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.PAL   and   PARAM.PAL   on
                                      unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.PAL   and   PARAM.PAL   on
                                      unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

000000 ERRORS

*1:LPS3.OBJ=0:PERPAR.LPS/P,LPS3.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS3.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS3.PAL
                                      on unit 0)

000000 ERRORS

*1:LPS4.OBJ=0:PERPAR.LPS/P,LPS4.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS4.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.PAL on unit 0)

0?   (Mount cassette with LPS4.PAL
                                      on unit 0)

000000 ERRORS

For AR11 support, assemble the ARD files instead of LPS files:

*1:ARD1.OBJ=PERPAR.ARD/P,LPS1.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.ARD on unit 0)

0?   (Mount cassette with LPS1.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.ARD on unit 0)

0?   (Mount cassette with LPS1.PAL
                                      on unit 0)

000000 ERRORS

*1:ARD2.OBJ=PERPAR.ARD/P,LPS2.PAL/P

0?   (Mount   cassette   with
                                      PERPAR.ARD on unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

PASS 2

0?   (Mount   cassette   with
                                      PERPAR.ARD on unit 0)

0?   (Mount cassette with LPS2.PAL
                                      on unit 0)

000000 ERRORS

*1:ARD2C.OBJ=0:PERPAR.ARD/P,PARAM.PAL/F,LPS2.PAL/P

O?   (Mount   cassette   with
                                            PERPAR.ARD   and   PARAM.PAL   on
                                            unit 0)

O?   (Mount cassette with   LPS2.PAL
                                            on unit 0)

PASS 2

O?   (Mount   cassette   with
                                            PERPAR.ARD   and   PARAM.PAL   on
                                            unit 0)

O?   (Mount cassette with   LPS2.PAL
                                            on unit 0)

000000 ERRORS

*1:ARD3.OBJ=PERPAR.ARD/P,LPS3.PAL/P

O?   (Mount   cassette   with
                                            PERPAR.ARD on unit 0)

O?   (Mount cassette with   LPS3.PAL
                                            on unit 0)

PASS 2

O?   (Mount   cassette   with
                                            PERPAR.ARD on unit 0)

O?   (Mount cassette with   LPS3.PAL
                                            on unit 0)

000000 ERRORS

Building a load module may now be accomplished by following the
instructions in the preceding paragraphs.

B.3   BASIC/PTS-11

B.3.1   LPS Support

The LPS support for paper tape is supplied in twelve binary tapes:

LPS0.OBJ	Required	LPS kernel module for LPS11 or AR11 and DR11-K
LPS1.OBJ	Optional	Analog to digital conversion for LPS11
ARD1.OBJ	Optional	Analog to digital conversion for AR11 and DR11-K

LPS2.OBJ	⎫	Real-time clock (60 Hz line frequency) for LPS11
ARD2.OBJ	⎪	Real-time clock (60 Hz line frequency) for AR11 and DR11-K
LPS2C.OBJ	⎬ One is required	Real-time clock (50 Hz line frequency) for LPS11
ARD2C.OBJ	⎭	Real-time clock (50 Hz line frequency) for AR11 and DR11-K
LPS3.OBJ	Optional	Digital input/output for LPS11
ARD3.OBJ	Optional	Digital input/output for AR11 and DR11-K
LPS4.OBJ	Optional	Display for LPS11 or AR11 and DR11-K
PTSSTR	One is	Patch for BASIC with strings
PTSNST	required	Patch for BASIC without strings

The following files are provided in source form:

FTBL.MAC	Function Table Module
PERVEC.MAC	Vector Definition Module
PTSINT.MAC	Interface Module
PERPAR.MAC	Parameter

To build a load module BASLPS (BASIC with LPS support), the parameter file PERPAR.MAC is edited and then assembled with FTBL, PERVEC and the appropriate interface module. The six object modules produced are then linked with the LPS and BASIC object modules to produce a load module. The specific instructions that are given to the system programs (editor, assembler, and linker) are given in the examples that follow the general description of load module building.

The two patch tapes, PTSSTR and PTSNST, alter one location in BASICL to permit the LPS scope to be refreshed by a background routine, a routine that is active while BASIC waits for input. A patch tape should only be used with BASIC/PTS V01. The Patch tape used should be linked after the last LPS tape. PTSNST should be linked when BASICL without strings is linked and PTSSTR should be linked when BASICL with strings is linked.

<div align="center">NOTE</div>

All of the procedures in this section assume that an unaltered PERPAR is being edited. It is recommended that a copy of the original PERPAR be made and saved for future use.

The BASINT interface module should be used with all versions of BASIC except BASIC/PTS V01 which should have PTSINT used in the place of BASINT.

For the LPS routines to be accessible from the BASIC CALL statement, the routine must be defined in a System Function Table. FTBL.MAC is a function table in source form. If any user-written assembly language routines are also linked with BASIC the routines must be defined in this function table.

PERVEC is the vector definition module. It defines the hardware addresses of the status registers and the interrupt vectors. The standard hardware address for the LPS interrupt vector is 340 (octal).

In PDP-11E10 machines with LPS support, however, the interrrupt vector is location 300 (octal). To assemble PERVEC with the interrupt vector at 300 (octal) it is necessary to delete the semicolon before the $V=0 definition in PERPAR.MAC. If the interrupt locations are at another location in memory then correct the interrupt addresses by using the system editor to define $V in PERPAR equal to the interrupt address minus 300 (octal). For example, if the LPS interrupt vectors start at 320 (octal) define $V=20 (octal).

To link the LPS module with BASIC, it is necessary to delete the semicolons (;) before the $LPS=0 statement. If any of the four optional modules are not to be included, a semicolon (;) must be inserted before the appropriate conditional.

Parameter	Insert ; before parameter if
$ADC=0	module LPS1 or AR11 is not to be included.
$CLK=0	module LPS2 (LPS2C) or AR11 is not to be included.
$DIO=0	module LPS3 or AR11 is not to be included.
$DIS=0	module LPS4 or AR11 is not to be included.
$STRNG=0	no string version of BASICL is to be linked.

Using the system assembler, the sources are assembled in the following combinations to produce the needed LPS object modules:

Object File	Source Files
FTBL	PERPAR,FTBL
PERVEC	PERPAR, PERVEC
BASINT or	PERPAR, BASINT
PTSINT	PERPAR, PTSINT

After these modules have been assembled, the LPS support may be linked with the BASIC object modules with only the desired optional LPS modules included in the LINK command strings.

Example Load Buildings

Examples are given for building a load module for the following three systems:

1. BASIC with strings, with complete LPS (or AR11) support, and with the LPS (or AR11) interrupt vectors located at 340 (octal).

2. BASIC without strings with complete LPS (or AR11) support and with the LPS (or AR11) interrupt vectors located at 300 (octal).

3. BASIC with strings, with a partial LPS (or AR11) configuration (one that includes the ADC, DIS, and CLK but does not include the DIO), and with the LPS (or AR11) interrupt vectors located at 340 (octal).

To build a load module called BASLPS.LDA, the following instructions should be followed:

Load ED-11 (the PDP-11 Paper Tape Software Text Editor) which is used to edit a source paper tape, PERPAR, which describes the required options. Place the supplied PERPAR.MAC in the high speed paper tape reader then follow one of these three procedures:

1. For a load module of BASIC including strings with support for a complete configuration of the LPS (all four optional modules) and with the LPS interrupt vectors at location 340 (octal), the following instructions should be given to the editor:

    ```
    *I H

    *O H

    *R

    *H
    ;$CAPS=0
    ;$CAPS=0
    *0A

    *D

    *H
    ;$LPS=0
    ;$LPS=0
    *0A

    *D

    *B

    */P

    *4T

    *
    ```

    At this point, the punch has the edited PERPAR.
2. Or for a load module of BASIC with no strings, with support for a complete configuration of the LPS and with the LPS vectors located at 300 (octal), the following instructions should be given to the editor:

    ```
    *I H

    *O H

    *R

    *H
    ;$CAPS=0
    ;$CAPS=0
    *0A

    *D
    ```

```
*H
$STRNG=0
$STRNG=0
*0A

*I
;

*H
;$LPS=0
;$LPS=0
*0A

*D

*H
;$V=0
;$V=0
*0A

*D

*B

*/P

*4T

*
```

At this point the punch has the edited PERPAR.

3. Or to build a load module of BASIC with strings for a partial configuration of the LPS, one that includes the ADC, DIS, and CLK but excludes the DIO with the LPS interrupts at 340 (octal), the following instructions should be given the editor:

```
*I H

*O H

*R

*H
;$CAPS=0
;$CAPS=0
*0A

*D

*H
;$LPS=0
;$LPS=0
*0A

*D

*H
$DIO=0
$DIO=0
```

```
*0A

*I

*B

*/P

*4T

*
```

At this point, the punch has the edited PERPAR.

Load PAL-11S. It is used to generate FTBL, PERVEC, and PTSINT object tapes. When PAL-11 has been loaded, put the edited PERPAR tape in the high-speed reader and answer the following questions.

```
PAL-11S

*S   H
*B   H
*L   P
*T   P
```

The PERPAR tape is now read.

EOF?   When EOF? is printed, remove the PERPAR tape and put the FTBL source tape in the high-speed reader. Then type carriage return.

END?   When END? is printed, remove the FTBL source tape and insert the PERPAR tape in the high-speed reader. When ready, type a carriage return.

EOF?   When EOF? is printed, remove the PERPAR tape and replace it with the FTBL source tape. (Note that both tapes are loaded twice in the assembly process.)

000000 ERRORS

PAL-11S

*S   When *S is printed, the new FTBL object tape is in the high-speed punch and may be removed. It is this tape (called FTBL) that will be used in the linking process to generate a new load module.

PAL-11S

These steps are then repeated using the PERPAR and PERVEC tapes to produce a PERVEC object tape and then repeated again using the PERPAR and PTSINT tapes to produce a PTSINT object tape.

Building the load module is accomplished by using LINK-11S. The bottom address specified should be 400.

Link the object tapes in the following order.

BASICL.OBJ

```
FPMP.OBJ
PTSINT.OBJ
PERVEC.OBJ
FTBL.OBJ
LPS0.OBJ
LPS1.OBJ or ARD1.OBJ
LPS2.OBJ or ARD2.OBJ (for 60 Hz line frequency)
LPS2C.OBJ or ARD2C.OBJ (for 50 Hz line frequency)
LPS3.OBJ or ARD3.OBJ
LPS4.OBJ
PTSNST
BASICH.OBJ
```

If $STRNG was left unchanged when editing PERPAR, BASICL with strings must be used. If a semicolon was inserted before $STRNG, BASICL for no string must be used.

Exclude the optional files for which a semicolon was inserted before the appropriate conditional in PERPAR. In the example given for a configuration not including the DIO, do not include LPS3.OBJ in the linking process.

To assemble the LPS from the sources the following procedure should be followed:

Load PAL-11S. It is used to generate LPS0, LPS1, LPS2, LPS3, and LPS4 binary tapes. Put the LPS0 source tape in the high speed reader. Enter the following commands

```
PAL-11S
*S  H
*B  H
*L  P
*T  P        The tape is now read.
END?         Re-insert the tape in the reader, press <CR>
000000

PAL-11S
*S
```

Repeat this procedure for the LPS1, LPS2(LPS2C), LPS3, and LPS4 source tapes or if the line frequency is 50 Hz, LPS2 should be assembled by the following procedure (LPS0, LPS1,LPS3, and LPS4 are still assembled as described above). Load ED-11 (the PDP-11 Paper Tape Software Editor) with the absolute loader, then create a parameter tape as follows:

```
*I  H

*O  H

*I
CYC50=0
        .EOT

*B

*/P

*4T

*
```

Load the PAL-11S assembler and insert the tape created by the above commands. Follow this procedure:

```
        PAL-11S
        *S   H
        *B   H
        *L   P
        *T   P
        EOF?        Insert the LPS2 source tape and press <CR>.
        END?        Insert the tape created by the editor and press
                    <CR>.
        EOF?        Insert the LPS2 source tape and press <CR>.
        000000

        PAL-11S
        *S          At this point the LPS2 binary tape is in the
                    high-speed punch.
```

The following table lists the assembly parameters for each module:

Source File	Conditionals	Define for Systems with:
LPS0.MAC	None	
LPS1.MAC	$AR11	AR11 hardware
	$CAPS	PTS hardware
LPS2.MAC	CYC50	50Hz line frequency (60 Hz is default)
	$AR11	AR11 hardware
	$CAPS	PTS hardware
LPS3.MAC	$AR11	AR11 hardware
LPS4.MAC	None	
VT55.MAC	VT55	VT55 terminal
Source File	Conditionals	Define for Systems with:
FTBL.MAC	$ADC	LPS1
	$CLK	LPS2
	$DIO	LPS3
	$DIS	LPS4
	$LPS	$LPS0 (all systems with LPS support)
	$VT11	VT11 support
	$DISK	RT-11
	$VT55	VT55 support
PERVEC.MAC	$LPS	LPS11 hardware
	$V	LPS interrupts not at location 340 (octal)
	$VT11	VT11 support
	$AR11	AR11 hardware
	$DR11K	DR11-K hardware
	$NUMBR=X	Multiple (X) DR11-K hardware
	OFFST1	First DR11-K interrupt address not at 167770 (octal)
	OFFST2	First DR11-K vector

| | | address   not   at   300
(octal) |
|------------|-------|-------------------------------------|
| PTSINT.MAC | $DIS | LPS4 |

To include AR11, simply define $AR11 in addition to $CAPS symbols in PERPAR source tape.

APPENDIX C

ERROR MESSAGES

This appendix summarizes the error messages that may occur when using BASIC Extensions call routines. See Appendix P of the RT-11 System Reference Manual for all error messages that may occur under the RT-11 system.

?ADC ERROR AT LINE XXXXX
      BASIC Ext Fatal   Cannot issue ADC command while an RTS operation is underway.

?ARG ERROR AT LINE XXXXX
      BASIC   Fatal   Arguments in a function call do not match (in number or in type) the argument defined for the function.

?BUF ERROR AT LINE XXXXX
      BASIC Ext Fatal   Buffer name given in LPS command has not been previously defined in a USE statement.

?DEV ERR-C
      BASIC Ext Non-fatal The device specified is illegal.

?DSP ERROR AT LINE XXXXX
      BASIC Ext Fatal   .DEVICE list space overflow. Redefine DSPSIZ in PERPAR.MAC and reassemble FTBL.MAC.

?DV0 ERROR AT LINE XXXXX
      BASIC   Fatal   Program attempted to divide some quantity by 0.

?NOR ERROR AT LINE XXXXX
      BASIC Ext Fatal   Number out of range.

?SYN ERROR AT LINE XXXXX
      BASIC   Fatal   The program has encountered an unrecognizable statement. Common syntax errors are mispelled commands and unmatched parentheses, and other typographical errors.

?UNF ERROR FUNCTION AT LINE XXXXX
      BASIC   Fatal   The function called was not defined by the program or was not loaded with BASIC.

GLOSSARY

Term	Definition
Analog	Numbers represented by directly measurable quantities (as voltages, resistances, or rotations).
Auto-gain	Software determination of the best switch gain value to use sampling analog data using the LPSAM-SG option.
Bipolar	Refers to a signal that is either positive or negative with respect to system ground.
Buffer	A temporary storage area which may be a special register or an area of storage.
Clock	A time-keeping or frequency-measuring device within the computer system.
DMA	Direct Memory Access. The DMA is attached to any PDP-11 allowing memory data storage or retrieval at memory cycle speeds without processor intervention.
Gain	An increase in signal power. Gain in the ratio of output power to input power.
Global	A value defined in one program module and used in others. Globals are often referred to as "entry points" in the module in which they are defined and "externals" in the other modules which use them.
Initialize	To set counters, switches, addresses and variables to zero or other starting values.
Library Routine	A collection of standard routines which can be incorporated into larger programs.
Mass Storage	Pertaining to a device such as a disk or DECtape which stores large amounts of data readily accessible to the central processing unit.
Overflow	A condition that occurs when a mathematical operation yields a result whose magnitude is larger than the program is capable of handling.
Parameter	A variable or an arbitrary constant appearing in a mathematical expression, each value of which restricts or determines the specific form of the expression.

Preamplifier   An electronic circuit or device that detects and sufficiently amplifies weak signals.

Source file   A file to be used as input to a translating program such as MACRO or BASIC.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems
        with software should be reported on a Software
        Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

_____
_____
_____
_____
_____
_____
_____

Did you find this manual understandable, usable, and well-organized?
Please make suggestions for improvement.

_____
_____
_____
_____
_____
_____

Is there sufficient documentation on associated system programs
required for use of the software described in this manual? If not,
what material is missing and where should it be placed?

_____
_____
_____
_____
_____
_____

Please indicate the type of user/reader that you most nearly represent.

☐ Assembly language programmer
☐ Higher-level language programmer
☐ Occasional programmer (experienced)
☐ User with little programming experience
☐ Student programmer
☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
                                                        or
                                                      Country

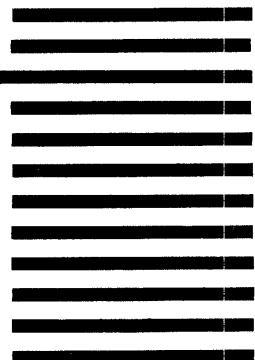If you require a written reply, please check here. ☐

Please cut along this line.

-- Fold Here --

--- Do Not Tear - Fold Here and Staple ---

digital

digital equipment corporation