The high-level programming language for on-line experimentation developed at Purdue University was built around the BASIC language and will be referred to as Purdue REAL-TIME BASIC (PRTB). The software was developed by modifying the BASIC Interpreter available from Hewlett-Packard (H.P. 20392). The modifications generated have involved the development of a series of machine-language subroutines which are directly callable from the BASIC software and which are designed to communicate in a variety of ways with experimental systems. The characteristics of the PRTB data-acquisition and control software are exactly analogous to those described in Sec. II.

As normally provided by Hewlett-Packard, the BASIC software for a stand-alone (single-user) computer does not include instructions for controlling data acquisition or other devices for on-line experimentation. These subroutines must be written by the user and added to the H. P. BASIC software.

In this appendix we will describe some general considerations for modifying the BASIC software available from Hewlett-Packard in order to generate a modified BASIC for on-line experimentation.

The discussions presented here apply to a paper-tape-oriented computer system for single-user operation. The specific details will vary somewhat depending on the total configuration of the system, the current version of H. P. BASIC used, and the nature and length of subroutines to be added. For an accurate treatment of the reader's particular system, the manufacturer's specific documentation and recommendations should be followed.

The linkage format and the steps involved in the execution of an auxiliary machine-language subroutine using the Hewlett-Packard BASIC language are illustrated in Fig. B-1. Each subroutine is designated by a unique integer number from 1 to 63,10. The CALL statement must first specify the subroutine number; subsequently, several variable or constant parameters may be included in the CALL statement. For example, if subroutine 5 is a data-acquisition segment requiring certain initialization information, like the number of data points to take, the clock frequency, and the address of the
first word of the data-storage buffer, these parameters could be transferred through the CALL statement as shown below:

\[
\text{CALL (5, } T, F, A(1))
\]

Before executing subroutine 5, **BASIC** will evaluate each parameter and set up an *address stack*. Upon entering the subroutine, the address of the last location in the stack will be in the \( A \) register. The stack itself will contain the