A programming system for doing asynchronous I/O on an 8K PDP-12*

MANARD R. STEWART

Northern Illinois University, DeKalb, Illinois 60115

A programming system is described for the PDP-12 computer system. The system handles input-output requests from user-written routines.

The purpose of the present paper is to discuss a programming system for the PDP-12. The system is designed to handle input-output (I/O) requests from user-written routines, thus relieving the programmer from much of the drudgery involved in coding programs that utilize the peripherals of the PDP-12. The system operates with the program interrupt (PI) facility enabled and provides flexibility that is not possible with systems that check device flags with the PI disabled. Program interrupts are handled in either PDP-8 or LINC mode, so user-written routines can execute instructions in either mode without causing problems.

The main advantage of the system relative to one not utilizing the PI facility is that I/O can proceed asynchronously with the mainstream program (at least from the user's point of view). This gives the user the capability of doing computation at the same time I/O is going on. It is also possible for more than one I/O device to be operating at the same time. For example, information can be output to the teleprinter while data is being read from LINCtape. Some examples demonstrating the capabilities of the system will be presented below.

Basically, the system consists of a collection of subroutines that perform specific I/O-related tasks. The subroutines provided by the system control I/O for the Teletype and LINCtape, but user-written routines for devices associated with the PI facility are easy to incorporate. In addition, several locations within the system can be altered to increase or modify the capabilities of the system. The Teletype-related routines and functions provide the basic monitor system; therefore, this part of the system will be discussed first.

TELETYPE MONITOR SYSTEM

Internally, characters are represented as trimmed 6-bit ASCII and are packed two per computer word. The internal code for a carriage return is 00 (octal), and all I/O to the Teletype is terminated with a carriage return followed by a line feed and an idle. Thus, the carriage

*The material covered in the present paper assumes the reader is familiar with a PDP-8 mode language or assembler such as DIAL, PAL, or MACRO-8. The DIAL assembler is described in the Digital Equipment Corporation LAR-6 DIAL Programmers Reference Manual (DEC-12-SE2D-D).

return has a special status in the monitor system. A number of additional keys have a special status within the system. The special keys will be discussed before detailing the methods of producing I/O with the system.

The rubout key can be struck in input mode to delete the last character typed and can be hit repeatedly to delete more than one character. In addition, a number of characters can be typed at the same time the control (cntl) key is depressed to accomplish various functions. Table 1 lists the characters echoed by the monitor and the functions resulting from typing them. The reader may note the similarity of the control functions to larger DEC machines such as the PDP-15.

The routines associated with the Teletype system are: TTYIN, TYPE, WAIT, and WAITR. How to input information from the Teletype to a user buffer can be understood from the following examples:

```

TTYIN = 35
WAIT = 152
.
JMS I TTYIN
BUFF
JMS Z WAIT
.
.
BUFF, 0
*.+23

```

In the example, information typed on the Teletype keyboard would be converted into 6-bit ASCII and packed two per word starting at Location BUFF. The JMS Z WAIT is included to insure that the subsequent code will not be executed until input is complete (i.e., a carriage return is typed). Location BUFF can be on any page of Field zero, and the statement *.+23 saves room for up to 40 characters, including the carriage return.

The example is set up with the AC assumed zero when the JMS I TTYIN is executed. If the AC is not zero, the

Table 1
Special Control Characters and Functions for
for the Keyboard Monitor System

↑ C	Reloads the dial system
↑ D	Reloads the dial system
↑ P	Transfers control to user-defined address
↑ S	Transfers control to location 400 of field 0. (400 is used as the starting address for most PDP-8 mode programs.)
↑ T	Same function as P, but can transfer to another address.
↑ U	Deletes the entire input line so it can be retyped

routine TTYIN expects one more argument to follow the buffer address. For example,

```

      .
      TAD CNTL
      JMS I TTYIN
      INFO
      ADDR
      JMS Z WAIT
      .
      .
      ADDR,  CLA
      .
      .
      INFO,   Ø
      .
      .+11

```

This example shows how to set up ↑P or ↑T to transfer control (see Table 1). If CNTL is positive and nonzero, then Control P would be set up. At any time after the special character is set up, striking the P with the control button depressed results in transfer of control to Location ADDR. If CNTL is negative, ↑T is set up and not ↑P.

Of course, the JMS Z WAIT in the previous examples could be left out; however, the code following the call to the input routine would be executed before the user could strike a Teletype key.

To type out information on the Teletype, the routine TYPE is used. The following example illustrates how this can be done.

```

      TYPE = 37
      .
      .
      JMS I TYPE
      MSG
      JMS Z WAIT
      .
      .
      MSG,  TEXT Z THIS IS AN EXAMPLE. Z
      .
      Ø

```

In this instance, the message "THIS IS AN EXAMPLE" would be typed on the teleprinter followed by a carriage return and line feed. The routine WAIT is used to wait until the message is output before proceeding. Note that a Ø was placed in the line following MSG.¹ Placing the Ø in the next line makes sure the string terminates with a carriage return.

The other wait routine (WAITR) can be used to repeatedly transfer control to a user subroutine while the Teletype is busy. The following example gives some indication of the generality of this option.

```

      JMS I TTYIN
      BUFF
      JMS Z WAITR
      DISPLY
      .
      .
      DISPLY, Ø
      .
      CLA
      .
      .
      .
      JMP I DISPLY
      .
      .
      BUFF,   Ø
      .
      .+23
      WAITR = 161

```

Suppose the subroutine DISPLY wrote the characters starting at Location BUFF on the scope. Further, the subroutine returned when the character represented by ØØ was encountered. The example would then display the characters as they were typed. The display would terminate on a carriage return. In other words, in using WAITR, program control is passed to subroutine display repeatedly until input or output terminates.

LINCTAPE INPUT AND OUTPUT

LINCTape I/O handling operates in block (nonfile) oriented manner. Both input and output are independently double buffered to maximize data rates in I/O transfers, and transfer of information to LINCTape proceeds asynchronously with mainstream programs.

The subroutines associated with LINCTape I/O are SEEK, READ, ENTER, WRITE, and CLOSE. The following examples should provide enough information on how they are used.

The first example shows how information can be read from LINCTape.

```

      READ = 176
      SEEK = 173
      .
      JMS I SEEK
      .
      7
      4ØØ
      JMS I READ
      ARRAY
      2Ø

```

This sequence of code reads 2Ø words into locations starting at ARRAY and ending at ARRAY+17 from Block 4ØØ of Unit 7 (all octal numbers). Any subsequent reads will continue from the next word on tape unless another JMS I SEEK is executed. Actually, the JMS I SEEK inputs two LINCTape blocks into the input buffer, and then the JMS I READ transfers 20 words

from the buffer to the array starting at Location ARRAY.

The second example shows how to write information on tape.

```

ENTER = 174
WRITE = 177
CLOSE = 176
JMS I ENTER
3
210
JMS I WRITE
ARRAY
10
JMS I CLOSE
DCA NFB
.
.
.
ARRAY, 1
        2
        3

```

The example would write 10 locations containing 1, 2, 3, etc., on Tape Block 210 of Unit 3. The JMS I CLOSE causes the output buffer to be padded with trailing zeros and the buffer to be written on tape. If the close was not included, subsequent writes would place the information starting with the next word on tape. If there was no close or write, the information would not be written on tape. The AC contains the next free block on returns from CLOSE.

Input and output may be mixed in any manner; however, when there is a mix, data rates may be slowed if only one unit is used (due to tape motion).

USER-DEFINED HANDLERS

The user may wish to code a handler for a device that is associated with the PI facility other than the Teletype or LINCtape. This section will provide information on how to add device handlers.

The following code shows how to incorporate a real-time clock (KW12-A) handler. The setup of the clock is done with a user subroutine.

```

EXIT = 123
CLKDIS = 124
CLOCK, 0
TAD HANDL
DCA Z CLKDIS
.
.
. (code to set up desired functions)
JMP I CLOCK
HANDL, TICK
.
.
.
TICK, CLA

```

```

.
.
. (handle interrupt from clock)
JMP Z EXIT

```

The first two instructions of the routine CLOCK place the address of where to go on a clock interrupt (i.e., TICK). On any interrupt caused by the clock, the program control is passed to Location TICK with the PI disabled. The final instruction, JMS Z EXIT, causes the interrupted program to continue its execution. It is not necessary to save and restore the link or AC in the user-defined handler.

The system skip chain to detect which device caused the program interrupt has room for four device flags, but can be extended easily.

LIMITATIONS

The entire LINCtape and Teletype system uses less than five pages of memory (640 decimal words), excluding LINCtape I/O buffers. Most of the page 0 is used, but Locations 2-17 (auto index registers) are available to the user. Since the basic system was designed to utilize as little core as possible, some limitations necessarily accrue.

The major limitation associated with the system involves the restriction that data and programmed entries must be in Field 0. However, there is no problem in having programs or routines which execute in other fields; the interrupt system will operate correctly. The limitation applies only to data and calls to the subroutines of the system.

The system also has a few bugs associated with the control characters. For example, typing cntl C occasionally reloads the dial system incorrectly or not at all. In addition, some of the control characters may produce incorrect results. However, these two bugs have trivial consequences and occur relatively infrequently.

Finally, the programming system has been in use for less than 1 year on one machine. As a result, additional bugs or shortcomings will become obvious in the future.

Used with other software, such as randomization, display, and real-time clock subroutines, a reasonably broad class of psychological experiments can be coded in a very short time (usually less than a day). In addition, with a few double-precision and conversion (BCD) routines, data analysis can be done utilizing the system. For example, a signal averager for some special data will run in less than 1 min. The same averaging task takes over 45 min with FOCAL.

NOTE

1. Actually, the line with 0 does not need to be included, because there is an odd number of characters in the string. The TEXT pseudo-op pads the last word half with 00 when there is an odd number. Thus, a carriage return is already there. However, counting characters is not very exciting, so using one additional word of core is usually worth the price of not doing a boring task.