T H E U N I V E R S I T Y O F M I C H I G A N

Memorandum

DEC 338 LIGHT PEN SENSE INDICATOR

Stephen F. Lundstrom

CONCOMP: Research in Conversational Use of Computers
F. H. Westervelt, Director
ORA Project 07449

supported by:

DEPARTMENT OF DEFENSE
ADVANCED RESEARCH PROJECTS AGENCY
WASHINGTON, D. C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

November 1966

ABSTRACT

This memorandum describes the characteristics and possible uses of the light-pen sense indicator that has been added to the Concomp Project's DEC 338 Display. The circuit involved, including the wire list used to implement this additional 338 feature, is also given.

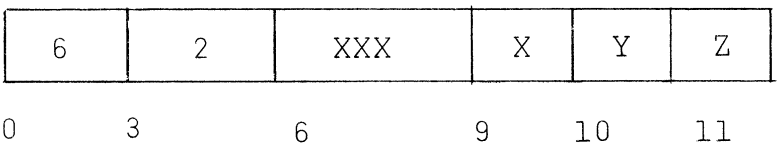
I. GENERAL DISCUSSION

The additon of the light pen sense indicator (LPSI) to the 338 Display allows the 338 Display Control to sense the past history of the light pen. The impetus for this addition came from programming considerations of some types of display program sequences. A later section describes some of the potential utility of this feature. In general, one may notice that there are many instances in display programming when the light pen is to be used to point generally at something (perhaps at a light button) rather than to indicate precise display coordinates. In many cases, the only reason for pointing is to move into another portion of the display file. Previously, this requirement was satisfied by PDP-8 central processing unit (CPU) interaction. With the addition of the LPSI, the display program itself may satisfy these requirements without CPU interaction.

II. LIGHT PEN SENSE INDICATOR (LPSI) DESCRIPTION

The light pen sense indicator is set on any light pen hit at any time; it may not be disabled. The 338 Display Control may, under program control, clear the LPSI and/or cause a display program skip if the LPSI is not set.

The programmer accomplishes the above tasks by microprogramming the remaining two bits of the control state instruction 62XX as follows:



Bit 0 3 6 9 10 11

During the execution of the above instruction the microprogrammed bits Y and Z have the following meanings:

Y = 0 Do not clear the LPSI.

Y = 1 Clear the LPSI.

Z = 0 Do not skip on LPSI test.

Z = 1 Do a display file skip (2 locations) if the LPSI is not set.

III. POTENTIAL USES

This section will describe briefly two possible uses of the new LPSI. No attempt has been made to be exhaustive, only suggestive.

A. Light Pen Tracking

One of the major problems in light pen tracking is in moving the tracking cross fast enough when the operator is moving the light pen at high speeds. The following high speed tracking algorithm is an extension of one successfully implemented by the author.

1. Clear the LPSI and enable the light pen.

2. Draw the normal tracking cross. Any light pen hits during this time are treated in the same manner as in standard tracking procedure with CPU intervention and cross repositioning.

3. If there have not been any light pen hits yet (skip if LPSI not set), draw the next vector of an expanding pattern around the cross. (Concentric diamonds, concentric squares, and spirals are all effective.)

If there is a light pen hit while one of the vectors of the expanding pattern is being displayed, the CPU uses the address of that point as the position of the tracking cross and the light pen is disabled.

Repeat 3 until a hit is made or the pattern is exhausted before moving to the next part of the display file.

The reader may imagine the number of internal display stops and CPU interrupts that would be required to implement such a tracking procedure without the LPSI.

B. Automatic Light Buttons

To demonstrate the potential in the use of automatic light buttons a hypothetical case of computer-aided instruction will serve as an example.

The computer-aided instructional system consists of visual instructional material (perhaps text, graphics, or diagrams) and related questions. The system might operate as follows:

Lesson 1 would be displayed. When the student is satisfied that he knows the material, he points the light pen at the "TURN PAGE" light button, causing a question and multiple-choice answers to be displayed. The student then would point at one of the answers (which also would act as a light button). If correct, another question or the next lesson would be displayed; if incorrect, Lesson 1 would be displayed again.

Note that none of the above procedures requires CPU intervention.

The light buttons used above would be implemented in the display program as follows:

1. Clear the LPSI.

2. Draw the light button.

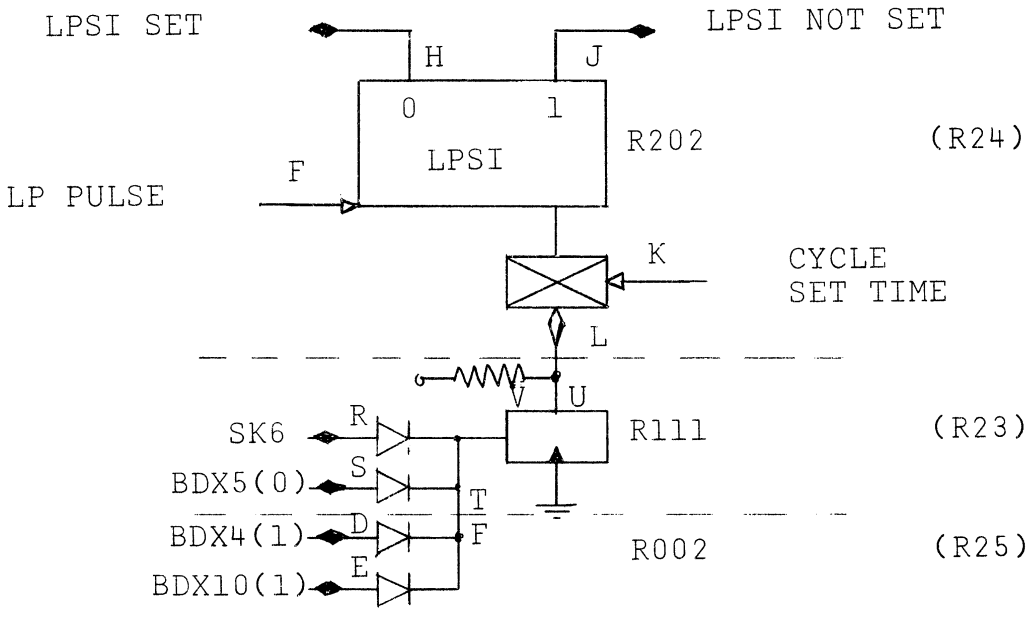
3. If the LPSI is not set, continue; otherwise execute a display jump to the appropriate place.

If CPU attention is required, an internal display stop may be executed, preferably following the rules set forth for calling PDP-8 subroutines from the display control. (see Appendix).

Again the reader will be left to determine the requirements on the PDP-8 CPU to implement a light button system as described above without the LPSI.

IV. IMPLEMENTATION

The light pen sense indicator was implemented with the addition of three modules to the 338 Display Control. The logic diagram for the LPSI is shown below.



Notice that the LPSI is set by the LP PULSE and is cleared if bit 10 is set in control state instruction 62XX at CYCLE SET TIME. This allows the sense of the LPSI to be used to condition the skip logic in the same instruction that clears the LPSI.

The skip logic is implemented by adding another legal skip condition to the R141 in U15. That condition is that the LPSI is not set at the same time that bit 11 of the instruction is set.

The following wire list was used to implement the LPSI.

DELETE:	U15R - U15C	GND
	U15T - U15V	GND
	U15R - U15T	GND
ADD:	U15T - U15V	GND
	U15P - R24J	LPSI NOT SET
	U15R - U6T	BDX11(1)
	S23K - R24K	CYCLE-SET TIME
	R23V - R24L	Enable Clear
	R23U - R23V	Enable Clear
	U10R - R23R	SK6
	U11K - R23S	BDX5(0)
	U11L - R25D	BDX4(1)
	U6R - R25E	BDX10(1)
	R25F - R23T	Gate
	P25R - R24F	LP PULSE

Insert the following modules at the indicated addresses.

R202	R24
R111	R23
R002	R25

APPENDIX

MEMO TO: ARPA Project Programmers

FROM: F. H. Westervelt

SUBJECT: Coding Conventions for 338 Display Internal Stop

The processing of internal stop interrupts generated from the execution of the STOP instruction (1400_8 in 330 control state) in a display file can be greatly simplified through the adoption of the following convention (shown in 8SS assembly code):

```
        DISPLAY FILE

        STOP            'STOP' HALTS DISPLAY (INT. STOP)
        SKIP            DAC STOPS HERE, RES 2 SKIPS 2 LOCS
        DC SERVICE      ENTRY ADDRESS FOR STOP SERVICE
        DC ARGMNT       OPTIONAL ARGUMENT

        FILE CONTINUES
```

The 'Internal Stop Interrupt Processor' then becomes very simple (and yet provides for arbitrarily complex processing of the internal stop):

```
*   INTERNAL STOP INTERRUPT SERVICE

ISIN    0           NORMAL ENTRY (WITH AC=0)
        RDAC        READ DISPLAY ADDRESS COUNTER
        DCA  AXR1   PUT INTO AUTO-INDEX LOCATION
        TAD* AXR1   GET ADDRESS OF SERVICE ENTRY
        SNA         SKIP IF NON-ZERO
        JMP* ISIN   IF ZERO, IGNORE STOP
        DCA  TMP1   PUT ADDRESS IN PAGE ZERO
        JMS* TMP1   GO TO SERVICE STOP
        JMP* ISIN   RETURN TO CALLER
```

If the called routine wants the optional argument, it should do a TAD* AXR1 to get it into AC.

In general, one expects the AC to be zero both going to and coming from service routines. Only when there is explicit agree-

ment between two routines to use the AC for transmission of data should the AC be non-zero. Failure to observe this convention results in the needless use of CLA instructions and waste of valuable memory locations.

14 November 1966 F.H.W.