27
6-4-76
25@NTIS

UCID-17130

Lawrence Livermore Laboratory

LIL8/V2: A LIST INTERPRETIVE LANGUAGE FOR THE MCS-8 MICROCOMPUTER

S. Bourret

September 19, 1974

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UCID-17130

REFER QUESTIONS TO: S. Bourret	LAWRENCE LIVERMORE LABORATORY — ELECTRONICS ENGINEERING **REPORT** LIVERMORE, CALIFORNIA — UNIVERSITY OF CALIFORNIA	**LER** 74-100901	
ORIG. S. Bourret	LIL8/V2: A LIST INTERPRETIVE LANGUAGE FOR THE MCS-8 MICROCOMPUTER	DATE 9/19/74	REV. 0
APPROVED Philip Siemens		PAGE 1 of 20	

1.0 INTRODUCTION

This paper describes a list interpretive language designed for microcomputer-based programmable control systems.

The purpose of this project was to create an easily understood method for scientific personnel (with minimal programming background) to program an MCS-8 microcomputer-controlled chemistry system, via teletype. The intent is for the language to compete in simplicity of understanding with a rotating cam programmer. Also needed was an interrupt capability, a special instruction set, and the ability to read and punch paper tapes of the user's programs.

In effect, this program simulates a different computer architecture resulting in an entirely new instruction set, I/O, and interupt capability. At present there are twenty-one instructions, the special interrupt option, and the ability to process sublists. New users' instructions are very easily added.

The approach taken was to use a code "word" for each specific input or output function (such as valve, sensor, relay, servo, etc.), and one or two words describing the specific condition for the function at the time. (Open, Close, etc.). The user builds his program in a list memory field, in the form of a table with the main input/output events listed in the order in which they are to occur. The program then steps through the table to perform the indicated operations. For example, suppose a user wants to open a valve called "Valve A" to fill a beaker with a chemical. Wait until a "full detector" signals, then shut off Valve A. Turn on a heater for five minutes, then open Valve B and wait until the gas is detected at a certain point, and then shut off all valves. The list of things that are to happen with a 15-word program would be as follows on Page 2.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Code word for selecting the output port that Valve A is on Binary bit pattern for "ON Valve A"	Turn on Valve A
Code word for a Wait Until instruction Binary bit pattern for Full Indicator input	Wait for beaker full indicator
Code word for output port containing Valve A, B, and the Heater Binary bit pattern for "OFF Valve A" and "ON Heater"	Turn off Valve A, Turn on Heater
Code word for Delay Octal word for number of seconds Octal word for number of minutes	Delay for 5 minutes
Code word for Valve B Octal bit pattern for "ON Valve B"	Turn on Valve B
Code word for Wait Until Binary bit pattern for gas detected input	Wait until gas is detected
Code word selecting output port containing Valve A, B Binary bit pattern for "OFF all Valves"	Turn off all valves

The programming, or list creating and changing, is done via a TTY and the control processing is achieved with an MCS-8 microprocessor. A program called ODT* is used to create the lists and start the program. Several sub programs to ODT (not given here) are available to manipulate the lists for changing, inserting, deleting and punching the lists on paper tape.

2.0 HARDWARE REQUIRED

The hardware required consists of the following: a microcomputer (Intel 8008 CPU and controls), an I/O interface with its input ports and receivers and its output ports and drivers, a TTY, the 256 word ODT PROM with its 256 word RAM, the LIL8 program, and the memory RAM fields set aside for the lists. A system block diagram is shown in Figure 1A, and a memory allocation map is shown in Figure 1B.

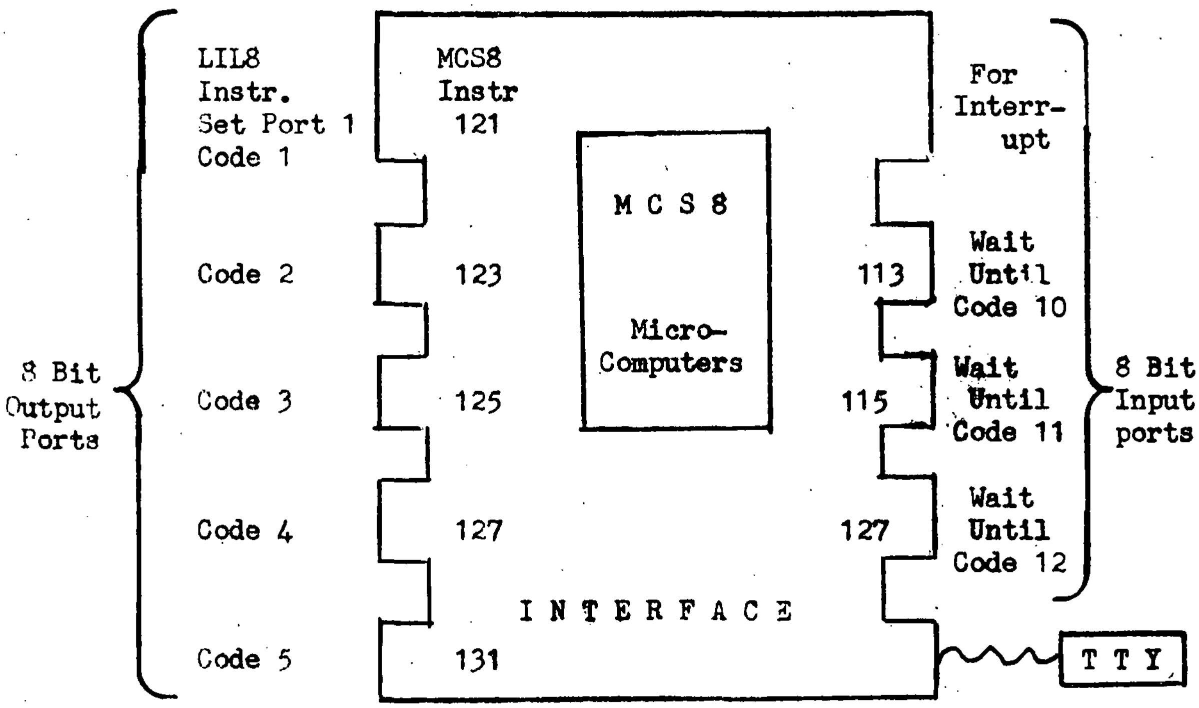


Figure 1A

LIL8/V2 HARDWARE BLOCK DIAGRAM

* Octal Debug Technique, by E. R. Fisher and J. C. English (See LER 72-103402)

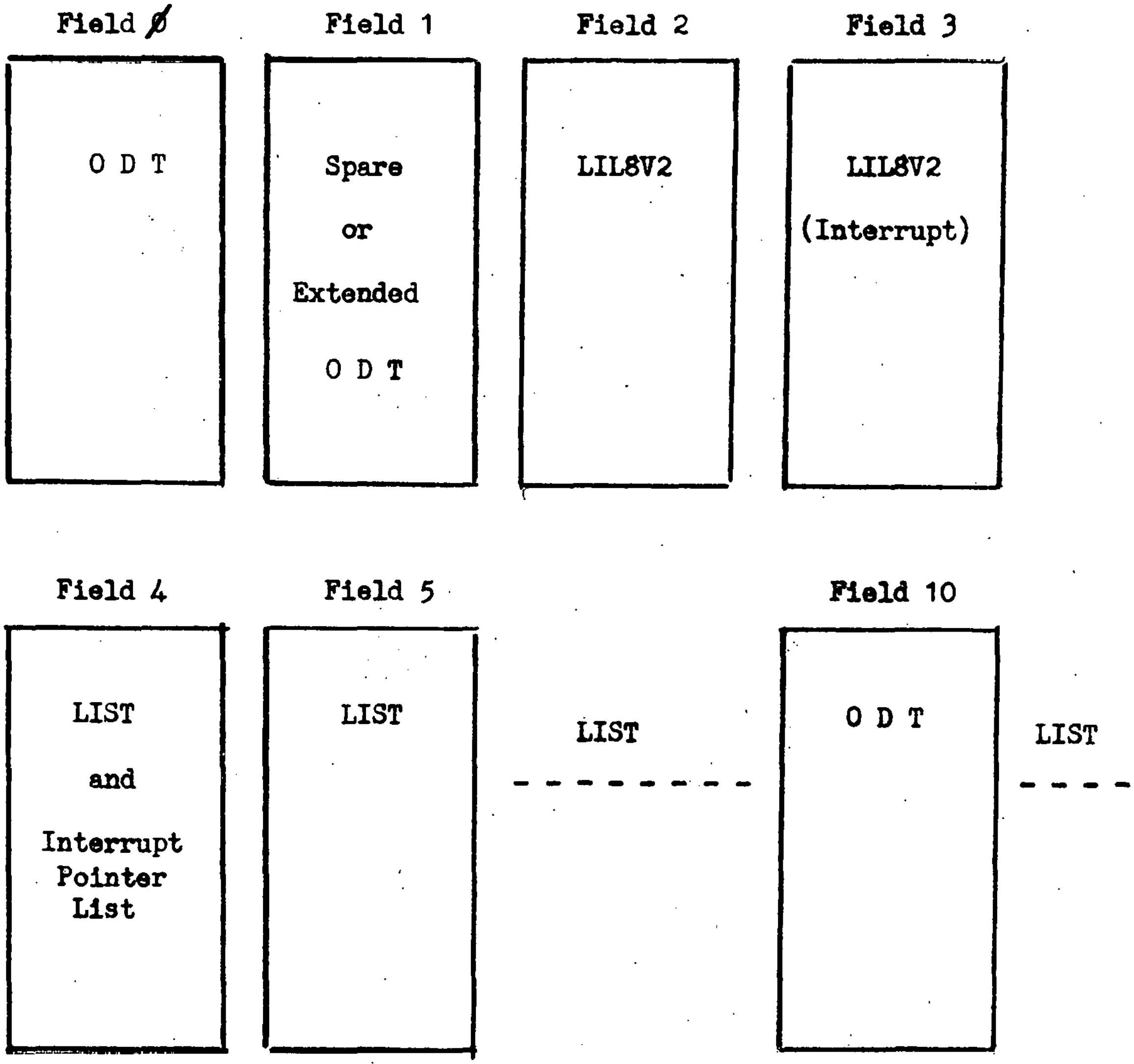


Figure 1B

LIL8/V2 SYSTEM MEMORY ALLOCATION MAP SHOWING FIELDS OF 256 WORDS

2.0 HARDWARE REQUIRED (Continued)

The LIL8/V2 provides for 5 output ports of 8-bits each, and 3 input ports of 8 bits, each. More ports could easily be added. For a better understanding of the hardware interfacing concerning I/O control signal assignment, refer to Table 1. Internally in LIL8/V2 an example of an MCS-8 instruction (not an LIL8/V2 instruction) given to output an 8-bit pattern in the A register, would be 123, meaning OUT2. (See Table 1).

In the memory allocation map, refer to Figure 1B, memory fields Ø and 1Ø are assigned to the ODT program. The LIL8/V2 program resides in memory field 2, and if the pseudo-interrupt option is used, it will take a small part of memory field 3. All other memory fields are available for the LIL8/V2 user lists. The lists can start and reside anywhere in the list memories, unless the pseudo interrupt option is used, and then a special list called the INTERRUPT POINTER LIST must begin in address ØØØ of memory field 4, and the list must end with a ØØØ. (See the INTERRUPT for more details.)

TABLE 1

LIL8/V2 I/O CONTROL SIGNAL ASSIGNMENT

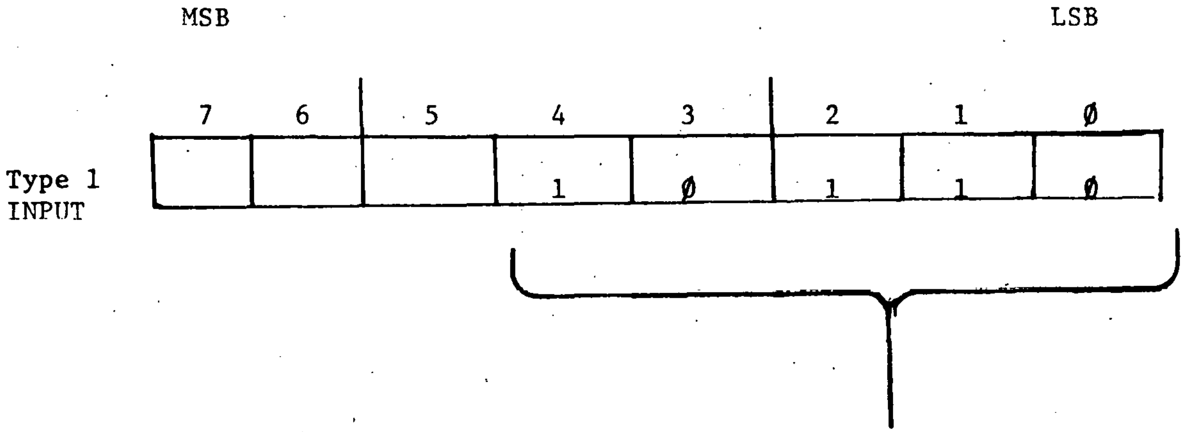
INPUT	MCS-8 INSTRUCTION	LIL8/V2 I/O CONTROL SIGNAL ASSIGNMENT
SEL Ø	1Ø1	
SEL 1	1Ø3	
SEL 2	1Ø5	TTY
SEL 3	1Ø7	TTY
SEL 4	111	INPUT PORT 1 PRIORITY
SEL 5	113	INPUT PORT 2 DIRECT 1
SEL 6	115	INPUT PORT 3 DIRECT 2
SEL 7	117	INPUT PORT 4 INTERRUPT

OUTPUT	MCS-8 INSTRUCTION	LIL8/V2 I/O SIGNAL ASSIGNMENT
SEL 1Ø	121	OUTPUT PORT 1
SEL 11	123	OUTPUT PORT 2
SEL 12	125	OUTPUT PORT 3
SEL 13	127	OUTPUT PORT 4
SEL 14	131	OUTPUT PORT 5
SEL 15	133	TTY
SEL 16	135	CLEAR
SEL 17	137	

3.0 DESCRIPTION OF THE CONTROL SYSTEM

The list interpreter has been selected to control as many as 5 output ports of 8 control bits each. These could be valve and/or motor controls. Each input to the system can be one of two types: type one is a 32-priority input which comes into an input port, and is encoded into 5-bits. (See Figure 2). The 5-bits allow 32 possible inputs and the priority insures order for one-at-a-time input.

The second type of input is a direct input of input ports. Here there are only 8 possible inputs/port, but they can be simultaneously input to the processor in one word.



Octal number 26 representing one of 32-priority encoded inputs.

Figure 2

3.0 DESCRIPTION OF THE CONTROL SYSTEM (Continued)

Delays are based upon the MCS-8 20µs instruction cycle timer. In LIL8/V2 the delays, minutes, seconds, and tenths of seconds are computed in software by the MCS-8, whereas in the forthcoming LIL8/V3, the delays will be computed by a hardware clock in order to relieve the system to handle multiple control tasks with the same processor. The following is the flow diagram for the LIL8/V2 program.

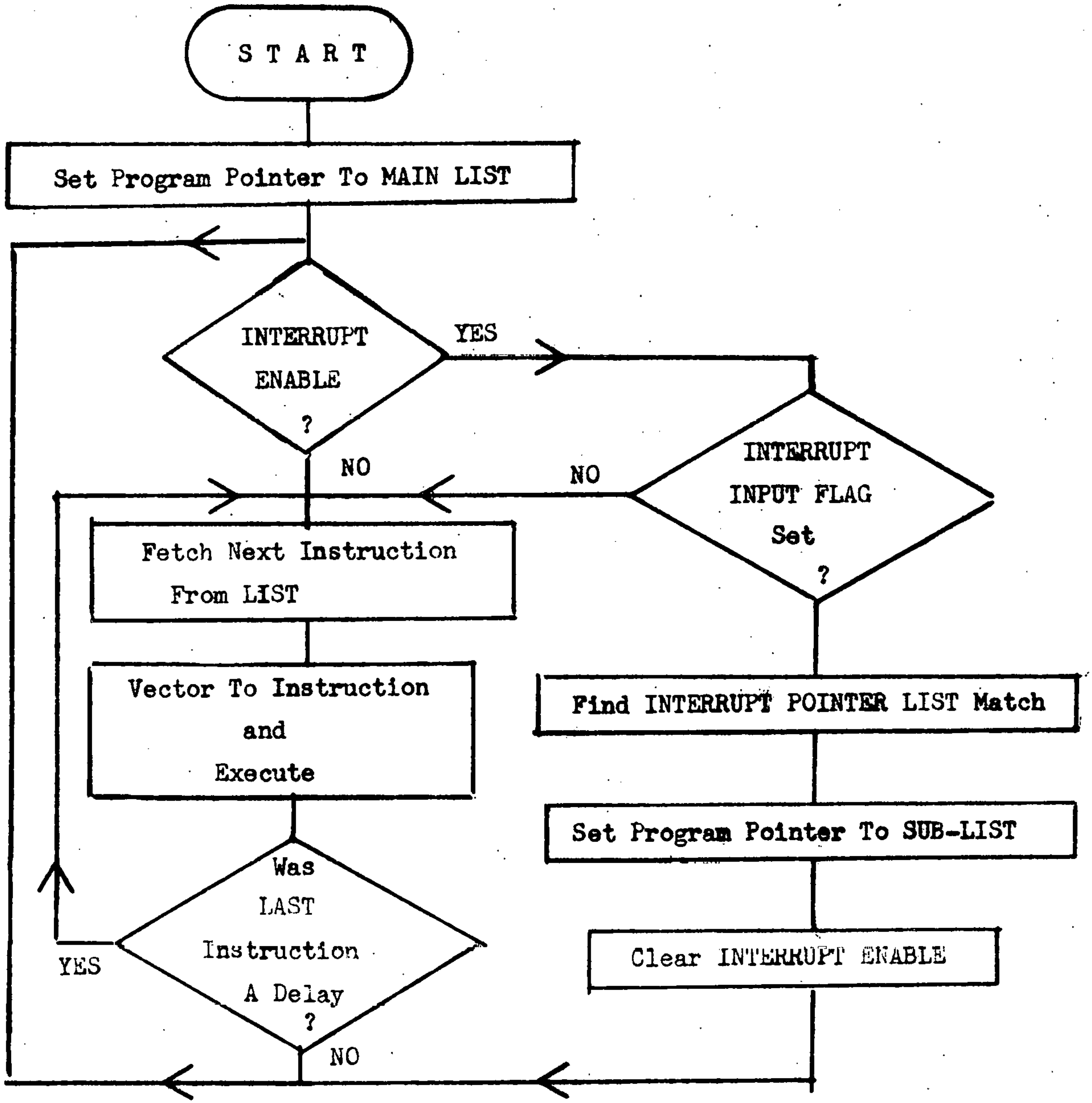

START
Set Program Pointer To MAIN LIST
INTERRUPT ENABLE ?
YES
NO
INTERRUPT INPUT FLAG Set ?
NO
Fetch Next Instruction From LIST
Vector To Instruction and Execute
Was LAST Instruction A Delay ?
YES
NO
Find INTERRUPT POINTER LIST Match
Set Program Pointer To SUB-LIST
Clear INTERRUPT ENABLE

Figure 3

LIL8/V2 FLOW DIAGRAM

4.0 DESCRIPTION OF COMMANDS

4.1 General

Each separate command has a code number. Some commands are followed by one or two words describing specifically what is to be done under that operation code.

4.2 Set Outputs

The codes for setting outputs are 1, 2, 3, 4, and 5. Each of the codes (from 1 to 5) refer to its output port of the MCS-8 hardware. In systems where the I/O is set up like memory reference instructions, this program could be easily adapted with minor changes. The following is an example of an output instruction:

ADDR	CONTENTS	
2Ø1	ØØ3	/Output port number 3
2Ø2	1Ø5	/Set the octal pattern 1Ø5 in the eight output registers of port 3

If the above example was controlling valves, the computer word to valve relationship would be:

7	6	5	4	3	2	1	Ø	Word bits
24	23	22	21	2Ø	19	18	17	Port 3 valves
Ø	1	Ø	Ø	Ø	1	Ø	1	= Octal 1Ø5

where,
Valves 1-16 are in ports 1 and 2,
valves 23, 19, and 17 would be turned ON by the 1Ø5 in the instruction and all other valves in port 3 would go OFF.

4.0 DESCRIPTION OF COMMANDS (Continued)

4.3 Delays

There are two kinds of delays. Delay 1 and Delay 2. Delay 1 is in minutes and seconds. This delay can last for as long as 256 decimal minutes and 256 decimal seconds. Delay 1 is a three-word instruction of code word 6, followed by the octal word for seconds, followed by the octal word for minutes.

Delay 2 is in tenths of seconds. This delay can last for 256 tenths of seconds. Delay 2 is a two-word instruction of code-word 7, followed by the octal word representing the number of tenths of seconds. Delay accuracy is based on 20 μs cycle time.

Example:

ADDR	CONTENTS	
1ØØ	ØØ6	/Octal code for Delay 1
1Ø1	Ø11	/Octal word representing nine seconds.
1Ø2	ØØ1	/Octal word for 1 minute.

NOTE: An interrupt cannot occur during, or immediately after a delay, thus, the interrupt check is not made until the instruction following the delay time is completed.

4.4 Input Instructions

There are two major types of input instructions: the WAIT UNTIL and the PSEUDO INTERRUPT.

4.4.1 WAIT UNTIL (WU)

The WU is a two-word instruction of code word followed by a specific input word. The code words for the three WAIT UNTIL instructions are 10, 11, and 12. Code 10 refers to the priority-coded input at port 1 and codes 11 and 12 refer to the direct inputs at input ports 1, 2 and 3 respectively. (See Table 1). The WAIT UNTIL instructions are used when further list processing is to be halted until input feedback from the controlled system matches the second word of the WU instruction.*

* The WU instruction can be interrupted by the pseudo interrupt.

4.4.1 WAIT UNTIL (WU) Continued

In the second word of the direct input instructions, only the bits that the instructions so designate by 1's will be checked, and the remainder will be ignored. For example:

ADDR	CONTENTS	
2Ø5	Ø11	/Direct input at input port number 2.
2Ø6	Ø2Ø	/Condition such that octal word Ø2Ø /must be met at input port number 2.

4.4.2 PSEUDO INTERRUPT

This is called PSEUDO INTERRUPT to distinguish it from the MCS-8's own built-in interrupt system which is only used for the restart button on the front panel. The purpose of the Pseudo Interrupt system is to allow certain predetermined inputs to cause a temporary halt of the processing of the main list and begin the execution of a sub-list at about any time during the process.* The portion of LIL8/V2 that contains the PSEUDO INTERRUPT was written in memory field 3, so that it could be left out if it wasn't being used.

There are three important points to the PSEUDO INTERRUPT: the ENABLE INTERRUPT instruction, the INTERRUPT SUB-LIST POINTERS, and the SUB-LIST.

4.4.2.1 ENABLE INTERRUPT (EPI)

The ENABLE INTERRUPT instruction is a one-word instruction of code word 13. When this instruction is given, a software flag is set, causing the program to check bit-7 of input port 4 each time before fetching the next instruction from the list. Bit-7 is a hardware flag indicating a change in the input of port 4. During an interrupt the software enable flag is automatically disabled.

4.4.2.2 DISABLE INTERRUPT (DPI)

The DISABLE PSEUDO INTERRUPT is also a one-word instruction of code word 14. This will clear the software INTERRUPT ENABLE flag.

* Interrupts are prevented during or directly after a delay.

4.4.2.3 SUB-LISTS

A SUB-LIST can be any part of the main program, or a completely separate part, but in order to return from a SUB-LIST to the point of interruption (or the next instruction after CALL) and continue on, the SUB-LIST must be ended with a RET (code word 15). If the interrupt system is to be active again after the INTERRUPT, then the ENABLE instruction must be given at the end of a SUB-LIST. (See Appendix).

4.4.2.4 RETURN (RET)

At the end of a SUB-LIST used by PSEUDO INTERRUPT, of the CALL instruction, there must be a way to return to the main list. The code number for the RETURN FROM SUB-LIST instruction is 15, and it is a single word instruction.

4.4.2.5 INTERRUPT SUB-LIST POINTERS

An INTERRUPT SUB-LIST POINTER is 3 octal words made up of an octal word expected at the INTERRUPT input port 4, followed by two more octal words of address and field representing the location of the SUB-LIST. The INTERRUPT SUB-LIST POINTERS must be placed, starting at the address ØØØ, in memory field 4, and ended with the word ØØØ. The SUB-LIST POINTERS are consecutively placed. For example:

ADDR	CONTENTS	
ØØØ	1ØØ	/In input port 4, bit-6 must be set.
ØØ1	237	/Location of SUB-LIST
ØØ2	ØØ5	/Field of above SUB-LIST
ØØ3	1Ø1	/In input port 4 bits 6 and Ø must be set
ØØ4	27Ø	/Location of second SUB-LIST
ØØ5	ØØ5	/Field of second SUB-LIST
ØØ6	ØØØ	/End of SUB-LIST POINTER list

4.4.3 How the INTERRUPT Works

The PSEUDO INTERRUPT works as follows: (Refer to Figure 3). After the ENABLE and the PSEUDO INTERRUPT instruction have been reached, then each time before fetching the next instruction from the list the most significant bit (bit-7) of input port 4 is checked to see if it is set. It it is not set, then the next set of instructions is fetched and executed as part of the normal sequence. It it is set, the INTERRUPT POINTER LIST (starting at address ØØØ in field 4) is checked for the combination that caused the interrupt. When it is found, then the program control is set to the address and field associated with the matching INTERRUPT POINTER combination and the INTERRUPT SUB-LIST (for that combination) is executed in the same manner as the primary list. In LPL8/V2 an interrupt cannot occur during a delay, as that could interfere with the user's timing.

4.5 NO OPERATION (NOP)

The NOP is a one-word instruction of code word 16. It gives one LIL8/V2 instruction delay.

4.6 BRANCH

The BRANCH instruction allows re-entry into any part of the program list. This instruction is comprised of three words: the code word 17 followed by the address of the re-entry point, followed by the memory field of the list. The instruction allows branching from one memory field to another. For example:

ADDR	CONTENTS	
263	Ø17	/Code word for BRANCH
264	26Ø	/Branch to address 26Ø, memory 4
265	ØØ4	/Memroy field 4

4.7 CALL

The CALL instruction is a means to utilize the same SUB-LIST over and over. As in the INTERRUPT SUB-LIST, this SUB-LIST is ended with a RETURN instruction. The same SUB-LIST can be used as an INTERRUPT SUB-LIST, except that this list is entered through hardware with the INTERRUPT. The CALL isntruction is a three-word

4.7 CALL (Continued)

instruction of code word 2Ø, followed with a word of address and a word of field. For example:

ADDR	CONTENTS	
267	Ø2Ø	/Code word for CALL
27Ø	237	/Address of SUB-LIST on CALL
271	ØØ5	/Memory field of SUB-LIST

4.8 END

At the end of the program list, the code word 21 must appear to end the program and return the program control to ODT.

5.0 PROGRAMMING THE LIL8/V2

The List Interpreter program can occupy one or two fields. If the INTERRUPT is used, it will take part of Field 3. The main program resides in field 2 and 3 and the programmable (RAM) program can reside in any available field. (See Figure 13). In addition to the LIL8/V2 program is the ODT program which is in field Ø (usually in (PROM memory) and field 1Ø, which is RAM memory. The user builds his program in available fields in the form of a table, or list of events, in the order that they are to occur. To gain access into a field using ODT, the user would type the field number, followed by an "S". For example: "4S" RETURN. The list is now ready to be created in any location of field 4.

To open a memory location in the field set by the "S" command (above), the user types the octal number of the memory location followed by a slash. The ODT program responds with the contents. For example:

4S↵ where ↵ is a carriage return

```
2/125
```

The contents of memory 4, address 2, is 125. To change the contents of an address, the user continues on the same line by typing the change, and then the carriage return.

```
2/125   252↵
```

To open the next consecutive address, the line-feed key would be substituted for the carriage return. For more information on using the ODT program see LER 72-103402.

6.0 IMPORTANT

The user must place the starting address of the main list and field in location 3ØØ and 3Ø1, respectively, in memory field 1Ø. This is where the program knows where to fetch the first instruction of the main list. This convenience was added, so that a portion of any list at a time could be checked out, and the list could be started at any logical point.

7.0 RUNNING THE LIL8/V2 PROGRAM AND USER LIST

If the correct list-building procedure has been followed, andthe starting address and field has been entered into memory field 1Ø, address 3ØØ and 3Ø1, then the program should run by the following ODT command:

```
2S ⤵

ØG
```

See following page for Example Program.

8.0 EXAMPLE PROGRAM

Here is an example program with contents and corrected errors. User types the underlined part.

```
                    ↓  Corresponds to "Linefeed"

                    ↵  Corresponds to "Return"

?                      /The Restart button sends program to ODT

4S ↵                   /Open memory 4

Ø1 Ø1Ø 5↓              /Open location Ø4 examine contents; it had a 1Ø,
                        I changed it to 5

ØØ4 ØD ØØ1 ↓           /First bank of solenoids, Code 1

ØØ4 ØØ2 125 252↓       /Change the pattern from 125 to 252

ØØ4 ØØ3 ØØ6 ↓          /Code 6 means delay 1

ØØ4 ØØ4 ØØ5 ↓          /Change from 5 seconds to 3 seconds

ØØ5 ØØ5 ØØØ ↓          /Ø Min

ØØ4 ØØ6 132 ØØ5 ↓      /5th bank of output valves, Code 5

ØØ4 ØØ7 Ø71 1↓         /Set ØØ1 pattern for output

ØØ4 Ø1Ø 12Ø 1Ø↓        /Priority WAIT UNTIL

ØØ4 Ø11 421 5↓         /Second word of WAIT UNTIL specifying a 5 for input

ØØ4 Ø12 3Ø1 21↓        /End

1ØS ↓                  /Change to memory field 3

3ØØ1 321 ØØØ ↓         /Starting address of User List Program

Ø1Ø 3Ø1 721 4↓         /Starting memory field of User List program

2S↵                    /Change to LPL8/V2 memory field

ØG                     /Start lsit processing

?                      /When the program is complete the END instruction
                       /sends the program control back to ODT
```

9.0 CONCLUDING REMARKS

The LIL8/V2 program makes a custom computer within an MCS-8 computer. It has its own set of instructions completely different from the machine language instructions of the MCS-8. Additional LIL8/V2 instructions are easily added to the program because access to an instruction is by a vector method, utilizing consecutively numbered instructions and a part of memory RAM 1Ø. Two additional instructions that could easily be added would be a DO instruction or a conditional IF instruction.

The main LIL8 program could be further reduced in length by using the absolute address of an instruction subroutine as the code word, but consecutive numbers make the code easier to remember and recognize.

See SUMMARY OF COMMANDS on the following page.

SUMMARY OF COMMANDS

SET (2 words)

CODE: 1, 2, 3, 4, 5

DELAY 1, DELAY 2 (3 words), (2 words)

CODE: 6, 7

WAIT UNTIL (WU) (2 words)

CODE: 10, 11, 12

10	Priority Coded
10,11	Direct inputs

ENABLE PSEUDO INTERRUPT (EPI) (1 word)

CODE: 13

DISABLE PSEUDO INTERRUPT (DPI) (1 word)

CODE: 14

RETURN (RET) (1 word)

CODE: 15

NO OPERATION (NOP) (1 word)

CODE: 16

BRANCH (BRCH) (3 word)

CODE: 17

CALL (CAL) (3 words)

CODE: 20

END (END) (1 word)

CODE: 21

APPENDIX

QUESTIONS AND ANSWERS

1. Q. How do you add new user instructions?

   A. First obtain the LIL8 source program. Now suppose the new instruction is to be called "IF", which would be instruction number 22, then insert a JMP;IF;MX instruction under the (RST/END CODE 21;RET TO ODT) instruction and then go off to memory field X and write the "IF" subroutine. The subroutine must be ended with the instruction JJZ;ENT;MA

2. Q. What is a memory field?

   A. Here, a memory field is called 256 words of memory. All the memory that can be accessed with one setting of the 8-bit H register.

3. Q. What happens if the interrupt is enabled before the end of the sublist is reached?

   A. There are no provisions for saving more than one return address at a time, so if a second interrupt comes along while serving the first, then the party of the first part would be forgotten.

DISTRIBUTION	NO. OF COPIES
G. G. Berg	1
S. C. Bourret	6
E. R. Fisher	1
E. A. Lafranchi	1
S. A. Nielsen	1
R. L. Peterson	1
P. D. Siemens	1
J. M. Spann	1
R. A. Thomas	1
EEIC	4
Patent Engineering	1

UCID-17130

LER 74-1009-01

DISTRIBUTION

LLL Internal Distribution:	
TID File	15
External Distribution:	
TIC	27
Robert Swartz Highland Park Illinois 60035	1

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy $____*; Microfiche $2.25

*Pages	NTIS Selling Price
~~1-50~~	~~$4.00~~
~~51-150~~	~~$5.45~~
~~151-325~~	~~$7.60~~
~~326-500~~	~~$10.60~~
~~501-1000~~	~~$13.60~~

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."

Technical Information Department
LAWRENCE LIVERMORE LABORATORY
University of California | Livermore, California | 94550