1985

IPCL1- An Interactive Process Control Language

Richard A. Erlandson
University of Central Florida

IPCL1 - AN INTERACTIVE PROCESS CONTROL LANGUAGE

BY

RICHARD ALLEN ERLANDSON
B.S.E.E., University of Illinois, 1963

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in the
Graduate Studies Program of the
College of Engineering
University of Central Florida
Orlando, Florida

Fall Term
1985

ABSTRACT

This report documents a Process Control Language. It was written to provide an easy-to-use, user-friendly language to control a manufacturing-type process. It is not assumed the user is proficient or even familiar with any computer languages. The user should be able to grasp the simple set of commands available and begin writing user programs in a short period of time. Emphasis has been placed on error messages to inform the user of the type of error and enough information to correct it. The language was written in PDP-11 assembly language and run on a 11/34 computer in the Microcomputer Laboratory at the University of Central Florida.

ACKNOWLEDGEMENTS

I wish to thank Mr. Clint Strange whose cooperation throughout, and continuation of, this work has been most gratifying.

I would also like to thank the faculty and staff of the University of Central Florida for sharing their expertise and knowledge. I would particularly like to thank Dr. Christian S. Bauer whose attitude and enthusiasm for his work has made a good part of this learning experience fun.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION TO IPCL1

Objective

The objective of IPCL1 was to develop a simple, easy-to-use, Interactive Processor Control Language (IPCL). The language was intended to be able to be used by someone not proficient in a computer language. IPCL1 was developed so that a non-computer professional could, with a minimum of training, write a program to control a simple process. As a follow-up on this project, Mr. Strange took the basic IPCL1 language, added disk storage and retrieval enhancements, along with other amenities.

Overview

The language developed for IPCL1 was a BASIC-type language. Some of the commands that the user has at his disposal are SET, IF, and GOTO which people familiar with BASIC will recognize. To keep the language as simple as possible the number of commands was kept to a minimum. To interface with the outside world a set of special input and output variables were defined and are under the control of the user.

Error messages are as descriptive as possible to aid the user in the correction of his/her error. For instance, if during the execution of a user program, a GOTO references a statement number that does not exist, the following error message is displayed: GOTO HAS NON-EXISTENT STATEMENT NO. Along with this error message the statment number of the offending GOTO statement will be given to show where the user program was aborted. As shown in Appendix E over 20 different error statements exist in IPCL1.

Also prompts are given whenever an input is expected from the operator giving him/her the options available at that time.

Variables defined by the user as well as input/output variables are easily displayed along with their present value using the DI(SP) command. Programmable delays during the execution of the user program are obtained via a single command, DE(LAY). The P(AUSE) command will halt execution of the user program for an unlimited amount of time until the user strikes the carriage return on the terminal at which time the program execution continues.

Any letters in a command shown in parenthesis are are optional, and need not be entered by the user when writing the program statement.

Comments can be displayed on the terminal during execution of the user program to aid in debugging the program. Statements can easily be deleted, K(ILL), or listed, L(IST), with these simple commands.

Memory Size

The program is written in PDP MACRO-11 Assembly Language. It consists of approximately 1000 lines of code using approximately 6500 bytes of memory. The basic IPCL1 also has a user program storage area of 16K bytes to hold the statements entered by the user. The 16K byte buffer will allow for approximately 200 user statements which should be more than adequate for most users. If more statements are needed it is only necessary to add another 16K bytes of storage area for each 200 added user statements.

Program Listing

While the IPCL1 program listing is not part of this report, it is available from the author and is fully commented and structured to make it as readable and maintainable as possible. A copy of the program, on floppy disk, will be kept in the University of Central Florida (UCF) microcomputer labs so that it can down-loaded and run with minimal effort.

Sample Program

A sample program, included in this report, demonstrates many of the commands available in IPCL1. A step-by-step analysis of the sample program is also included to show some of the possibilities using IPCL1. Flow diagrams and syntax diagrams of all the commands are also included in this report.

IPCL1 is a simple, user-friendly language that someone with minimal training can put to good use.

CHAPTER II

DESCRIPTION OF IPCL1

IPCL1 is a simplified, "user-friendly" interactive processor control language. A manufacturing or similar process can be controlled through a set of simple English-like instructions. Diagnostics have been included throughout the program to inform the user that an incorrect input has been made. Every attempt has been made to inform the user in non-ambiquous terms to aid in correcting the input. Some errors cannot be detected until the user program is actually running. When these types of errors are detected the user program is aborted, and the location in the user program that the error was detected is displayed along with an error message to pinpoint the type of error. Examples of error detection will be covered later when the format for the user program is given. Syntax diagrams in Appendix A are included to aid in in understanding the various areas of the program.

User Options

When IPCL1 is run initially the user has one of four choices:

1) RUN - this command tells IPCL1 to go to the first (lowest number) user statement and start executing the user program. The user program statements are interpreted by the IPCL1 program one at a time and executed sequentially unless the user has a GOTO XXXX command which would cause the statement XXXX to be executed instead of the sequential statement following.

2) LIST - this command lists the user's program starting at the lowest-numbered statement. If more than twenty statements exist in the user program the List portion of the IPCL1 program waits for the user to strike a key on the keyboard so program statements do not roll off the top of the screen before they can be analyzed by the user. LIST displays "NO USER PROGRAM" if no user program has been entered.

3) EDIT - this command allows a new user program to be entered or an existing user program to be modified.

4) TEST - this command tests all of the relays on the Gardner Box by operating them one at a time in approximately one-second intervals.

Editor

The EDITOR in IPCL1 is used to create new user programs or to change existing user programs. The user program consists of a sequence of program lines. A program line consists of an unsigned, four digit statement number followed by a statement. A comment following the statement may be added. A typical program line is:

```
1020  SET A = 1 ;Initialize  Variable A = 1
```

The ; and comment following is optional and does not affect program execution but is useful in debugging when using the listing of the program. Program statement syntax is explained in more detail later in the text and is supplemented by Syntax diagrams which appear in Appendix A.

The statement number is the first part of the program line. It must be a four digit number. If a statement number less than 1000 is used, the user must enter enough leading zeroes to make it a four digit statement number. The user program will be executed in numerical order, however the program statements do not have to be entered in numerical order.

Using the Editor

To call the EDITOR type in an E while in the MONITOR mode. When the EDITOR mode is entered an opening message is displayed informing the user of the options available while in the EDITOR.

To list a user program while in the EDITOR mode, type in an L. The user program will be listed sixteen lines at a time. To list the next sixteen lines, type a carriage return. An "End of File" message will be displayed when the end of the user program has been reached. If the user types an L and a user program has not been entered, the "No Program in Buffer" message will be displayed.

New statements can be inserted or added to the user program by choosing an appropriate statement number. It is strongly advised to use statement numbers at least ten apart when entering the original user program to leave room for later modifications to the program. If more statements are needed between two existing consecutive statements and there are not enough statement numbers available (statement numbers must be integer) then a GOTO statement can be used. The GOTO will direct the execution to a new sequence of statements with a GOTO at the end of the sequence of statements to direct the execution back to the desired statement.

Statement XXXX can be deleted from the user program by typing KILL XXXX where XXXX represents the statement number. If XXXX does not exist, an error message will be displayed.

To change an existing statement YYYY re-type the statement number and new statement. The new statement will replace the old one. For example, if the statement:

```
1020 SET B < A
```

exists, type the new statement:

```
1020 SET B = A
```

to change it or type:

```
KILL 1020
```

to delete it.

Refer to the SYNTAX diagrams in Appendix A for syntax of the various statements. These Syntax diagrams will show the exact structure of the program statement. As mentioned earlier, any characters of the command shown in parenthesis are optional, and need not be entered by the user. Refer to Appendix C for a sample program listing.

The last statement executed should be an END statement. Upon execution of this statement the message "END OF PROGRAM" is printed and control returned to the MONITOR.

To exit the EDITOR, type an E. This returns you to the MONITOR mode where the user program can be run. The user program can also be listed by typing an L. To return to the EDITOR from the Monitor, type an E.

When the user program is run, execution will begin at the lowest-numbered statement number and be executed sequentially unless a GOTO statement is encountered. The original user program will probably be entered in sequence, however, the statements can be entered in any order. When a listing is made, they will be shown in the correct sequence. For example, if the following statements were entered in this order:

```
0100 SET A = 0
1050 DIS A
1040 SET A = A + 1
1060 PAUSE
1200 END
0010 DIS ALL
```

The program would be listed in this order:

```
0010 DIS ALL
0100 SET A = 0
1040 SET A = A + 1
1050 DIS A
1060 PAUSE
1200 END
```

Program Statements

The format for the program statements available to the user are explained in the following paragraphs. For a synopsis of the formats see the Syntax Diagram in Appendix A.

The IPCL1 program language consists of the following commands:

```
;XXXXXXXX
PAUSE
ABORT
END
GOTO XXXX
ZAP
SET
DISPLAY
DELAY XX
CLEAR
IF
```

This set of commands will allow the user to construct a program which will be able to monitor and control I/O devices controlling a process. As seen in in the sample program described in Chapter III, a fairly complex operation can be controlled in relatively few program statements.

Using these statements the user can create a program to run and monitor a manufacturing process or similar operations. A sample of such a program is explained later in the text. The flow diagram for the sample program is in Appendix B. The listing for the program is in Appendix C and the schematic diagram in Appendix D.

Each statement is entered or altered while in the EDIT mode of IPCL1. Each statement is preceded by a four digit statement number. The statement number determines the order in which the statements will be executed while in the RUN mode. IPCL1 starts executing statements at the lowest-numbered statement and then continually executes the next statement in numerical order unless a GOTO is encountered or an END statement is reached.

If a GOTO statement is encountered the program execution will then continue at the statement number referred to in the GOTO.

Statements will continually be executed until an END, PAUSE, or ABORT statement is encountered. If an error is detetected during the running of the user program, the program may be aborted or halted, depending on the severity of the error. An error message, and the program statement that the program stopped on, will be displayed for the user.

;XXXX

This command allows a comment to be displayed during the execution of a program. For example, if the user wanted to know every time a certain area of the program was executed, a comment statement could be inserted. Every time that statement was executed, the comment would print out on the CRT terminal.

Comments placed after an executable statement are listed when a LIST of the program is made but not printed during program execution. Liberal use of these comments is recommended to aid in reading and debugging the program later. See the sample program listing in Appendix C.

PAUSE

This command is provided to halt execution of the program for an undetermined amount of time. The user continues execution by hitting the carriage return (CR) on the terminal. When this statement is executed the statement along with the statement number is displayed to inform the user where he/she is in the program. The display also tells the user to hit the CR to continue. This delay can used to make some needed adjustments on the equipment being controlled by the user-program.

ABORT

This command stops program execution and sets all output bits to 0. It is intended to be used when the user wants to stop the program and turn off all the outputs. After execution of this statement the user is back in the MONITOR mode of IPCL1 where he/she has the choice of Listing the program, Editing the program, Running the program, or running the Test program. A message is printed after the execution of this program informing the user that the program has been aborted.

END

This command stops program execution but leaves all the outputs in their final state. After execution the message "End of Program" and the END statement with its statement number is displayed to inform the user where the program stopped.

GOTO XXXX

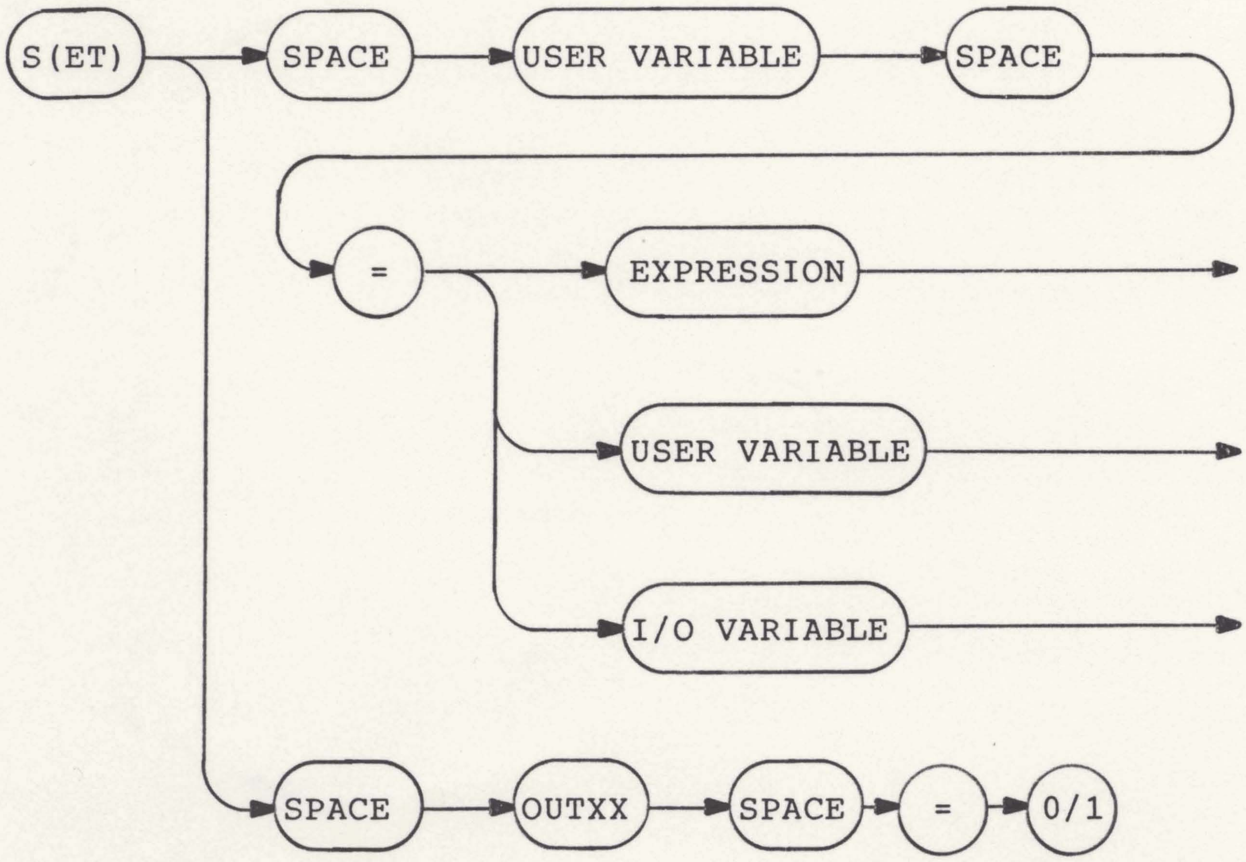
This command allows the user to start executing another statement other than the next one following the one presently being executed. After this statement is executed the next statement executed is the one specified (XXXX). Sequential execution of statements then continues until another GOTO statement is entered or the program stops.

ZAP

This command turns off (sets to 0) all of the output bits controlled by the user program.

SET

This command allows the user to set a user variable (defined by the user) to a value determined by an expression, another user variable, or an I/O variable. It also allows the user to set an output variable to 0 or 1. The syntax of the SET command is as follows:



A user variable is defined when used in the SET statement for the first time. To define a variable called CNT the following SET statement would be used:

```
SET CNT = 10
```

Thus, the variable CNT would be defined and have an initial value of 10. To increment the CNT variable use the statement:

```
SET CNT = CNT +1
```

To set the CNT variable equal to another pre-defined variable A, use the statement:

```
SET CNT = A
```

If variable A has not been defined at run time an error message will be printed and the program aborted.

The user variable can have up to four characters in its name. The first must be an alphabetic character. If a user variable with more than four characters are used an error message will be displayed at run time.

I/O Variables

I/O variables are input/output variables. Sixteen bits are available for output and sixteen available for input. The output variables are labeled OUT00 through OUT15. The input variables are labeled IN00 through IN15. INXX and OUTXX are reserved labels for I/O variables and cannot be used as labels for user variables.

To set bit 0 of the output variables to 1 use SET OUT00 = 1. Since the output bits can only be 0 or 1, only the value 0 or 1 can be used in the SET statement. Similarly any output variable can be set to a 0 or 1.

The input variables cannot be set as they are inputs from the Gardner (or similar I/O) box. These can be displayed or compared as will be described later.

DISPLAY

This command is used to display variable information during execution of the user program. Any variable can be displayed singly, or all of the variables can be displayed with one command. To display all of the variables, including user variables and I/O variables use the statement DISPLAY ALL. The user variables would be displayed first with their current values, then the I/O variables and their values would be displayed. To display only the input or output variables, use the statement DISPLAY IN or DISPLAY OUT respectively. To display a single variable use the DISPLAY statement with the variable's name following, e.g., DISPLAY OUT02 would display the value of output variable OUT02.

If the statement asks to display a variable that has not been defined in the user program up to the point it is to be displayed, an error message "NO SUCH VARIABLE IN TABLE" along with the statement will be printed.

DELAY XX

This command is used to halt the execution of the user's program for a given number of seconds. To halt the program for ten seconds the following statement would be used: DELAY 10. When this statement is executed there will be a ten-second delay until the following statement is executed. A delay of up to 99 seconds can be used the DELAY statement. If a number larger than 99 is used, the following error message will be printed: "DELAY TIME SHOULD BE LESS THAN 100 SECONDS." If no time is entered, "NO DELAY VALUE IN DELAY STATEMENT" will be printed.

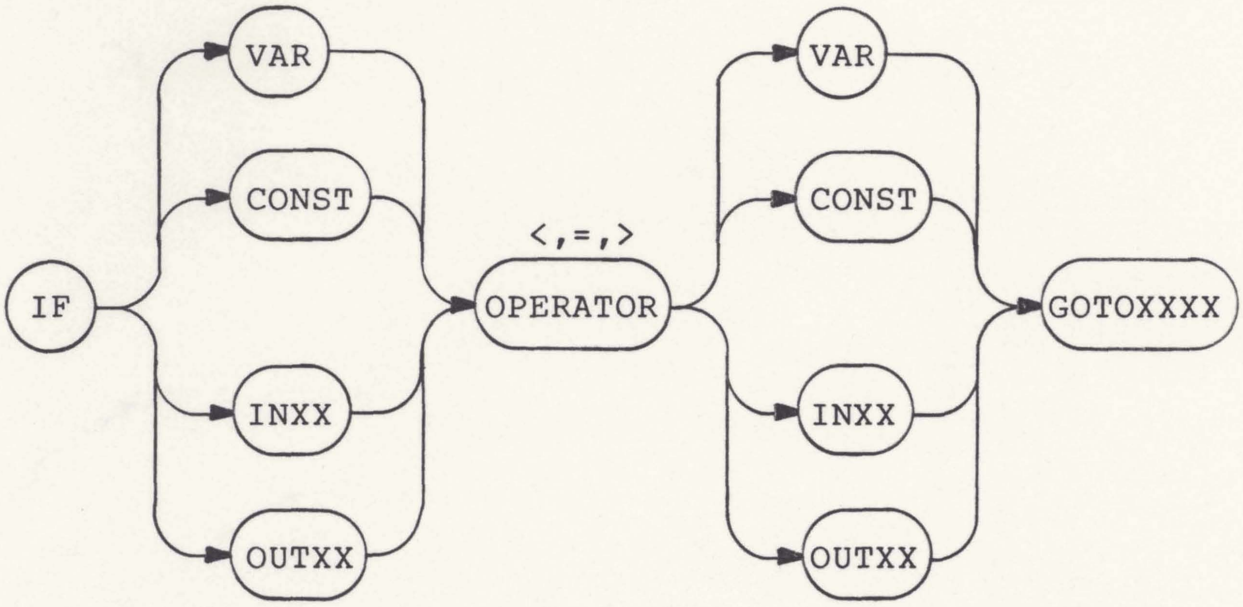
CLEAR/CLEAR ALL

This command is used to clear user-defined variables from the variable table. The statement CLEAR ABC would remove the variable ABC from the variable table.

The command CLEAR ALL would remove all of the user-defined variables from the variable table. I/O variables cannot be removed using the CLEAR statement. If an I/O variable is named in the CLEAR statement the following error message will result: "CANNOT CLEAR I/O VARIABLES FROM TABLE."

IF

This command is used for conditional branching. The syntax for an IF statement is:



The program will start executing the statement number in the GOTO XXXX portion if the values of the expressions on either side of the operator are such that the expression is true. For example, if A=1 and B=1 when "If A=B GOTO 1020" is encountered, the program will jump to the statement numbered 1020 and start executing because A = B. If, later in the program, A=1, IN00=0 and "IF A = IN00 GOTO 1050" is encountered, the program would not take a jump but execute the following statement because A did not equal IN00.

If a non-existent statement number is used in the GOTO XXXX portion of the statement the program will be aborted and an error message printed.

Three operators can be used in the IF statement. They are the =,>,or< sign. If another operator is used, an error message will be printed at execution time.

CHAPTER III

SAMPLE PROGRAM SYNOPSIS

The sample program is an operating program simulating a machine shop type of operation. The flow diagram is in Appendix B, and complete listing for this sample program is contained in Appendix C. The statements beyond the semicolon (;) are comments to aid the user to follow the flow of the program. Statements with only a comment after the statement number are printed out during the execution of the program to aid the user to determine when key points (determined by the user) are reached.

The interface to the computer (PDP-11) is attained through a Gardner box. The Gardner box provides sixteen bits of output and sixteen bits of input accessible to the computer via an interface board installed in the PDP-11. Each of the 16 output bits operate a relay in the Gardner box. The contacts of the relays are accessible and can be wired to operate the various motors needed to move the drill, turn the carrousel, and run the conveyor to simulate the machine shop operation. The schematic diagram is Appendix D.

These output bits are labeled OUT00 thru OUT15 inclusive. Similarly 16 input bits are provided that can be read by the computer. These inputs can be wired to switches to monitor events in the system. For example, the closing of switch SW1 indicates the pusher has reached its full reverse or "home" position. This switch is wired such that an input bit changes from a 0 to a 1 upon the closure of the switch. The input bits are labelled IN00 thru IN15 in the user sample program. The SW1 switch is connected so that IN01 becomes a 1 when SW1 is closed. The sample program monitors IN01 in a tight loop (only executing that command that reads IN01). When the IN01 bit turns to a 1 the program "knows" that the pusher is in the "home" position.

The sample program waits for an item to come down the conveyor belt, pushes the item onto the carrousel, turns the carrousel on to move the item under the "drill," turn on and lower the drill to drill the item, raise and turn off the drill, and deliver the finished item into a hopper by turning on the carrousel. If a new item was available within 5 seconds after an item was drilled it would push a new item onto the carrousel before turning on the carrousel and then start the drilling operation on the new item when the new item was in position.

The first statement in the program begins the initialization portion of the program by turning off all motors. The program then checks to see if the drill is in its full up or "home" position by checking to see if IN04 = 1. If not, it turns on the Up-motor for the drill until it is in the "home" position. The program then checks to see if the carrousel is in the home position by checking if IN03 = 1. If not, it turns on the carrousel motor until the carrousel is home. The program next checks to see if the pusher is in its full reverse or home position by checking IN01 to see if it equals 1.

The program then turns on the conveyor motor by setting OUT00 to 1. It then monitors switch OS1 (IN00) to determine when an item arrives. When an item is sensed, the conveyor is turned off and the item pushed onto the carrousel. The forward limit of the pusher is sensed by monitoring IN09. When IN09 equals 1 the forward pusher motor is turned off and the reverse pusher motor turned on. Again, IN01 is monitored to detect the pusher reaching its home position.

After the pusher is home the carrousel is turned on by setting OUT03 = 1. The program delays for one second before monitoring IN03 to give the carrousel time to move from its home position. When IN03 = 1 the item is under the drill and ready for the drilling operation.

The drill is turned on and the down-drill motor turned on. When IN05 = 1 the drill is all the way down, and therefore the down-drill motor is turned off. The drill is left down for five seconds, then raised by turning on the up-drill motor by setting OUT05 = 1. When the drill is fully up in its home position the up-drill motor and the drill are turned off.

The processing of the item is now completed. The conveyor is now turned on for five seconds to see if a new item is ready to be processed. If a new item is present after five seconds the item is pushed onto the carrousel and the process repeated. When the carrousel is turned on to position the new item under the drill the previously processed item is delivered into the hopper.

If no item is available after the conveyor has been on for 5 seconds the conveyor is turned off and the carrousel turned on to deliver the last item into the hopper. As before, a one-second delay is inserted after the carrousel is turned on before the program monitors whether the carrousel is in the "home" position. This delay allows the carrousel to "leave" the present home position after the carrousel motor is turned on.

At this time the conveyor is turned back on and the program waits for a new item. However, the program also monitors IN15. If IN15 = 1 (by manually operating switch SW6) the program turns off all motors and ends. Control is then back in the monitor where you have the normal options, i.e., LIST, EDIT, or RUN the program again.

CHAPTER IV

SUMMARY

IPCL1 is a user-oriented, friendly, high-level lanquage. It should be able to be mastered quickly. It is a good "starter" language for someone unfamiliar with computer languages in general.

This report is written with the intent of it being a "users manual" in that most of the emphasis has been given in describing the language and how to use it. It has been run and fully tested in the microcomputer laboratory where many of the error messages were developed.

The sample program alluded to in the text has been in operation in the laboratory controlling the operation of the various devices described. It was found to be very easy to change the operation of the "shop" through the use of the Editor and the simplicity of the language. The usage was made even more "user-friendly" by the addition of program storage capabilities by the continuation of the work by Mr. Strange.

APPENDIX A

IPCL1 SYNTAX DIAGRAMS

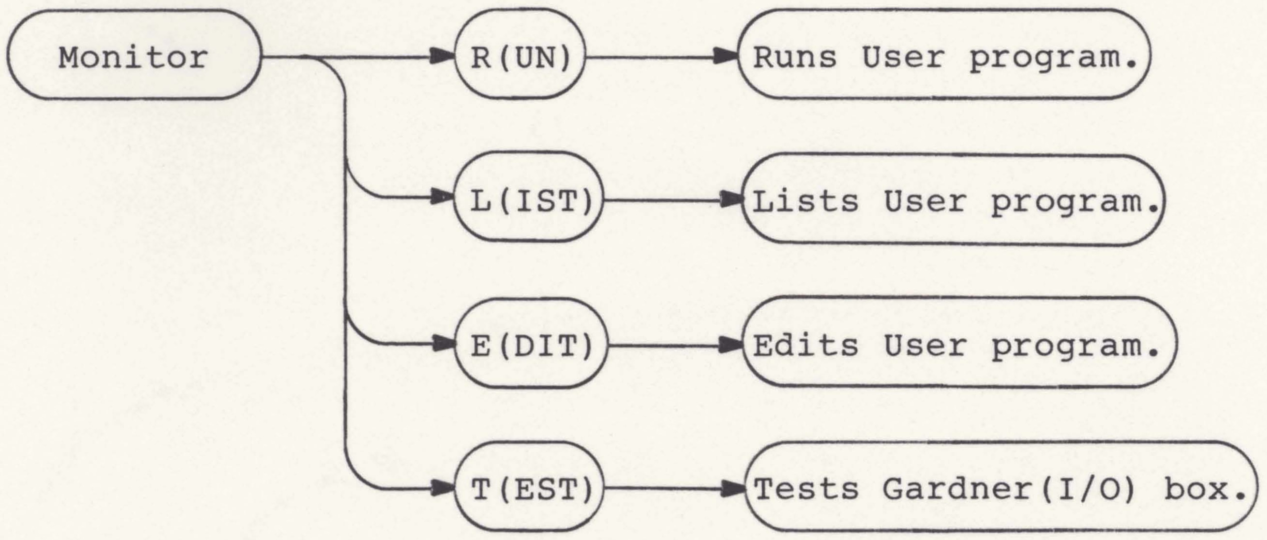


Figure 1. Monitor syntax diagram

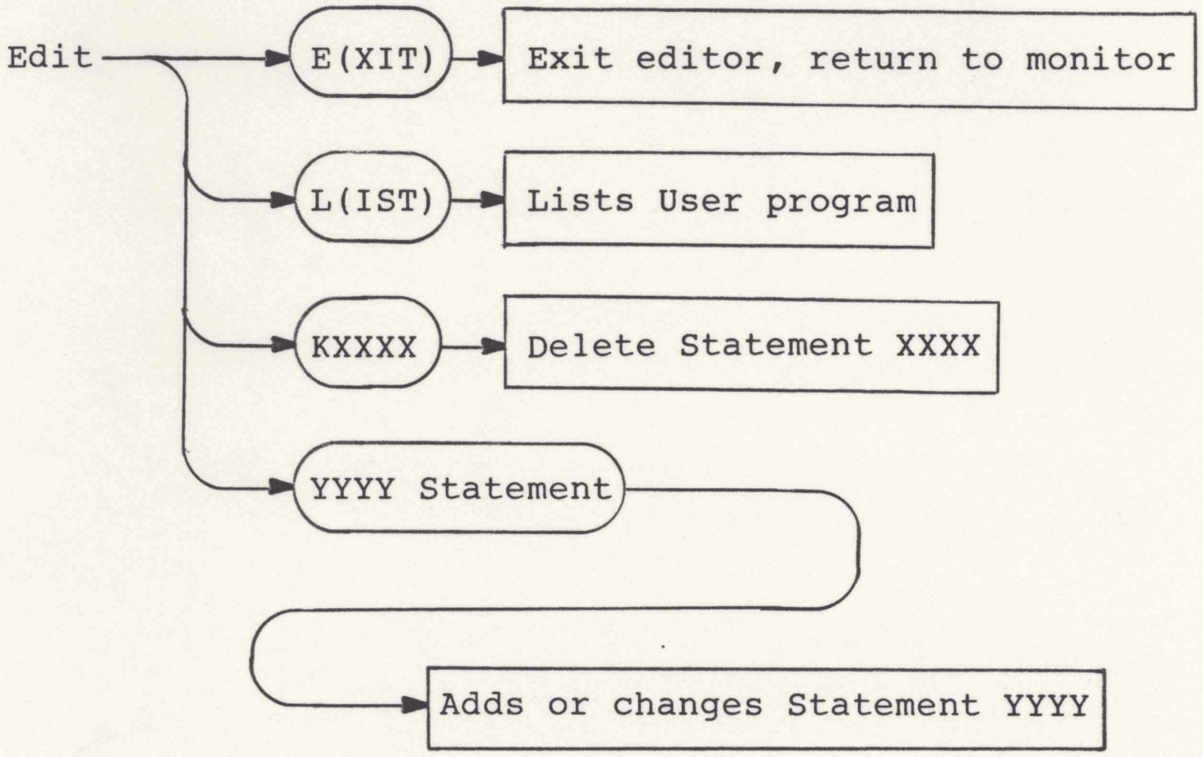


Figure 2. Edit syntax diagram

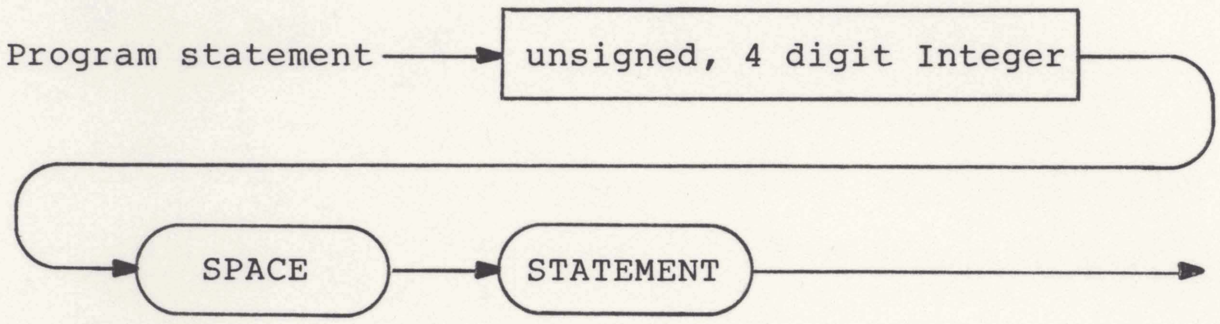


Figure 3. Program statement syntax diagram.

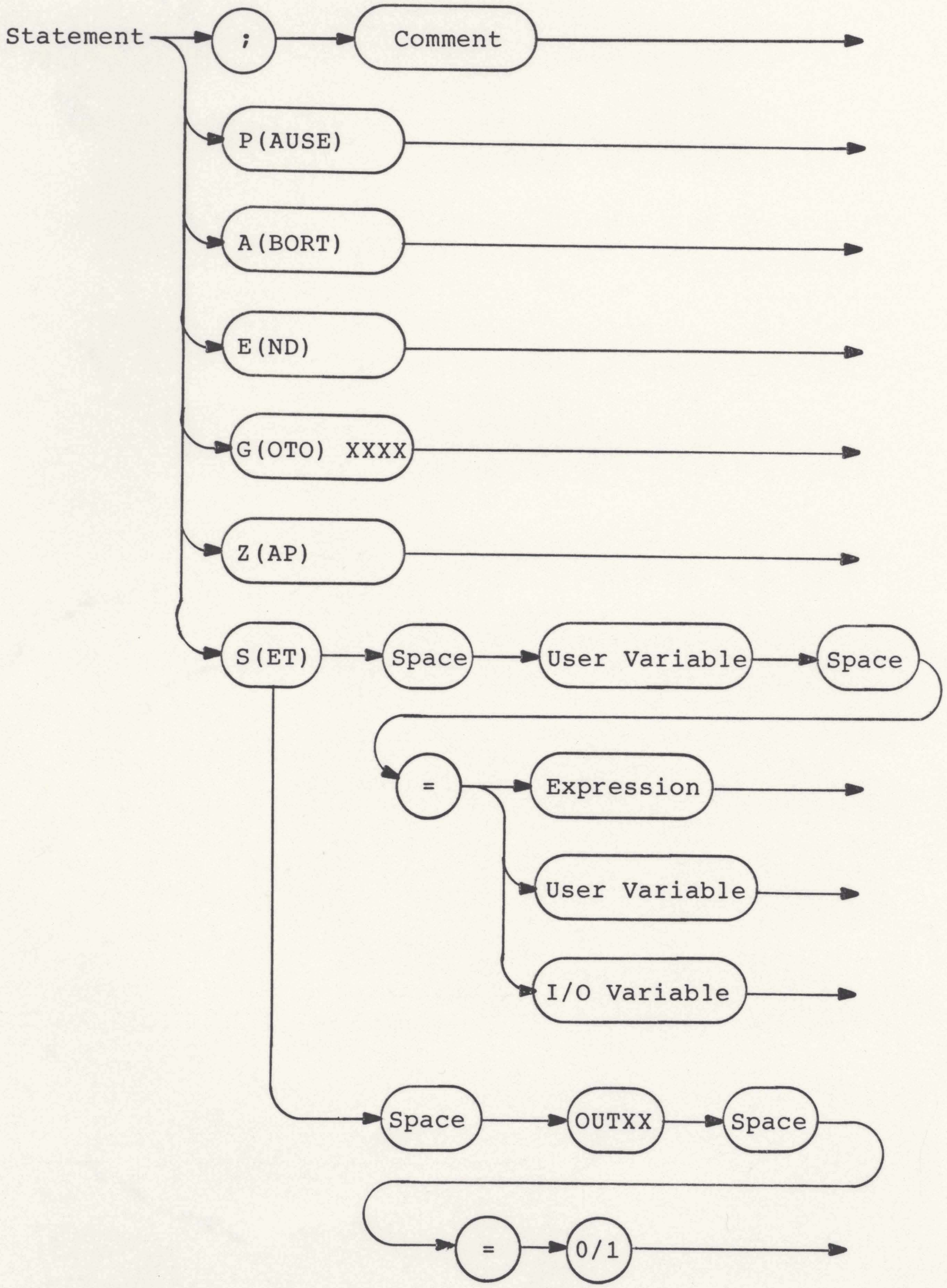


Figure 4. Statement syntax diagram

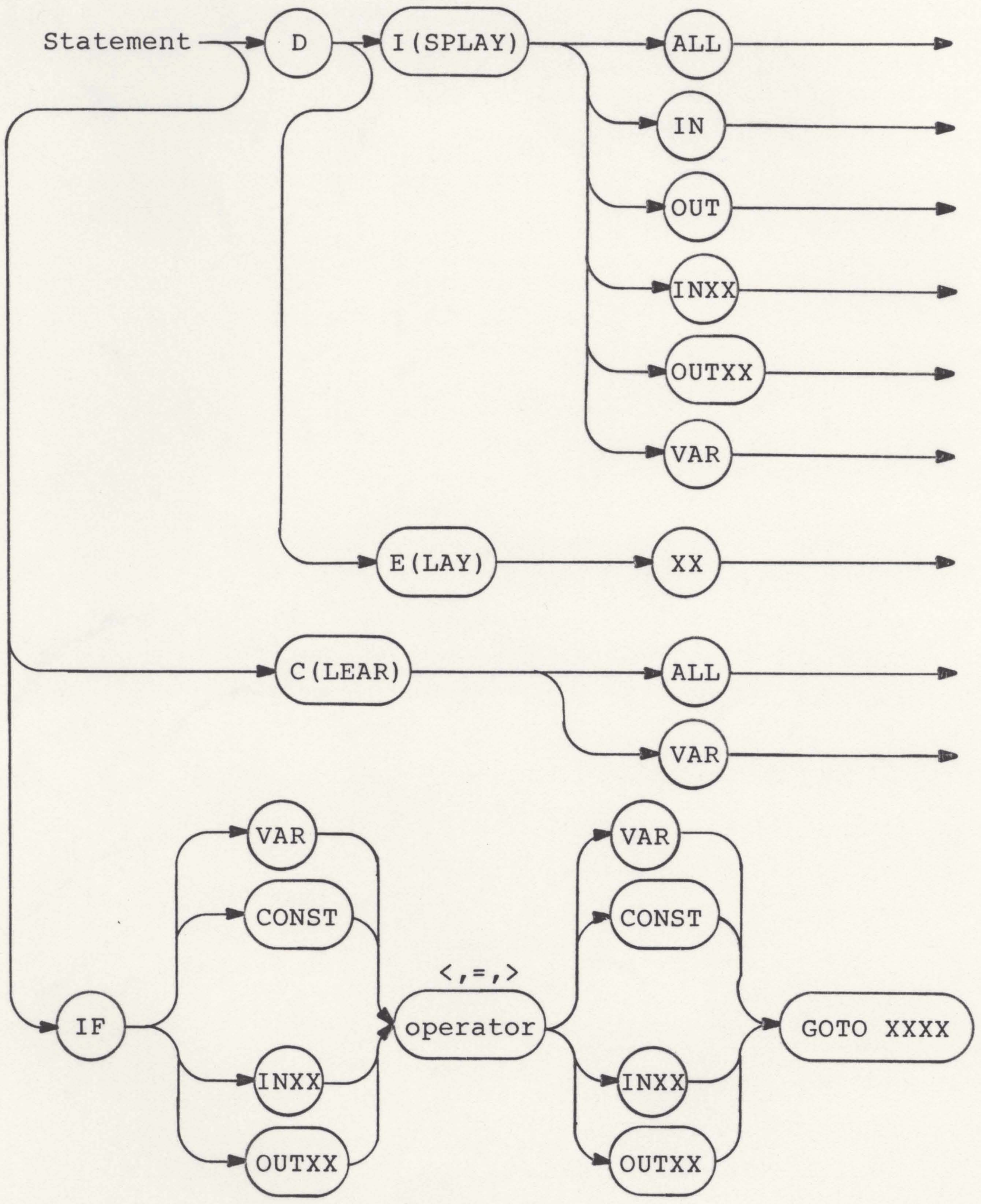
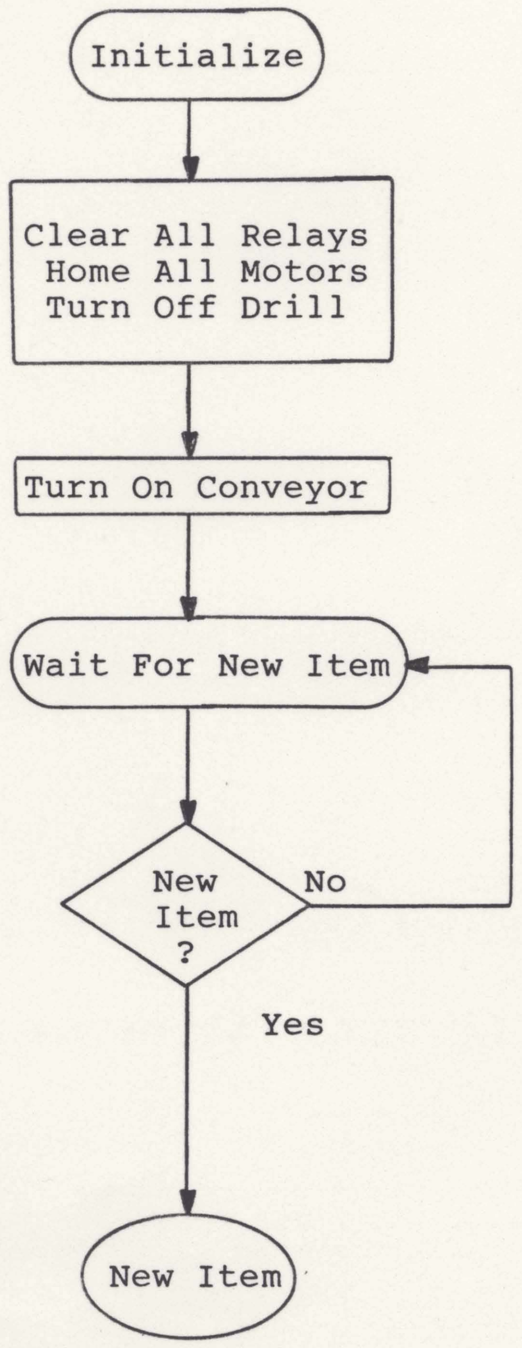
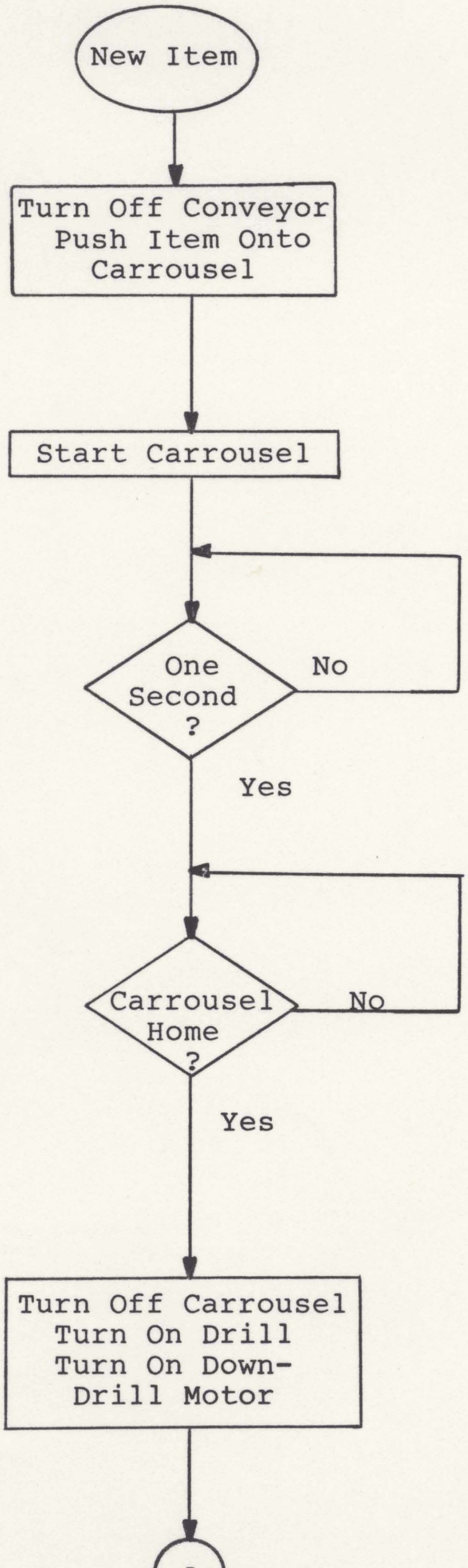


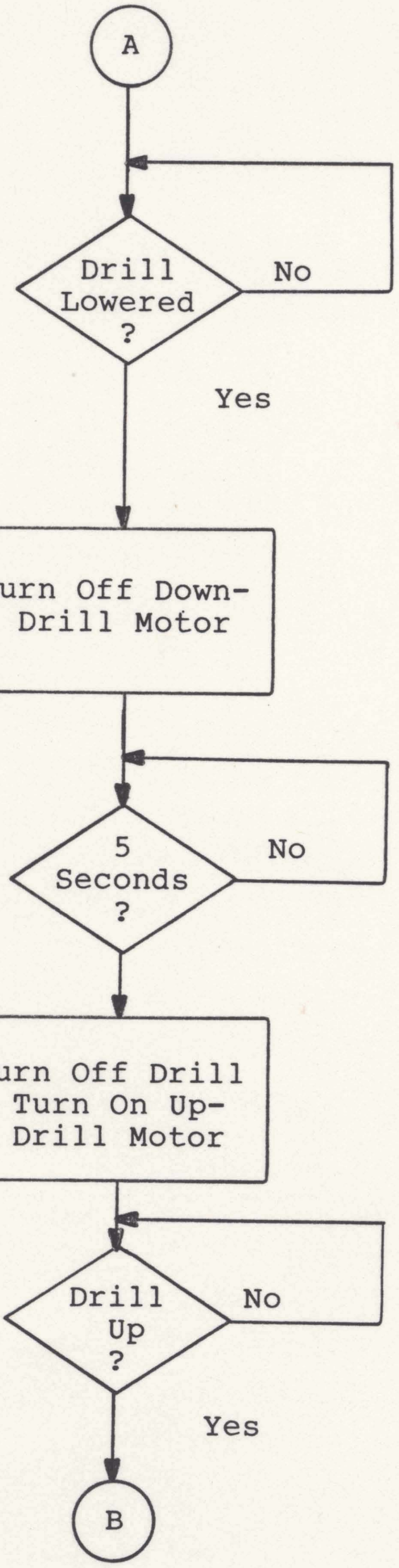
Figure 4. continued

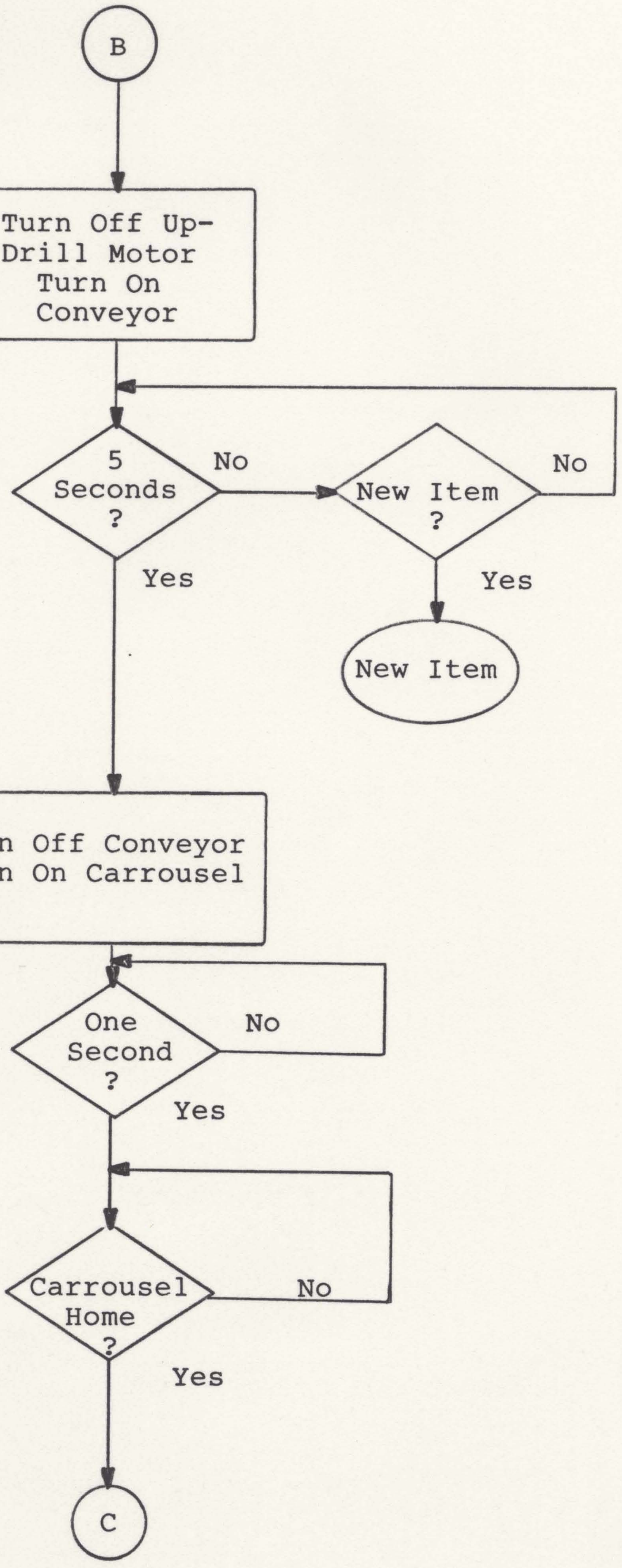
APPENDIX B

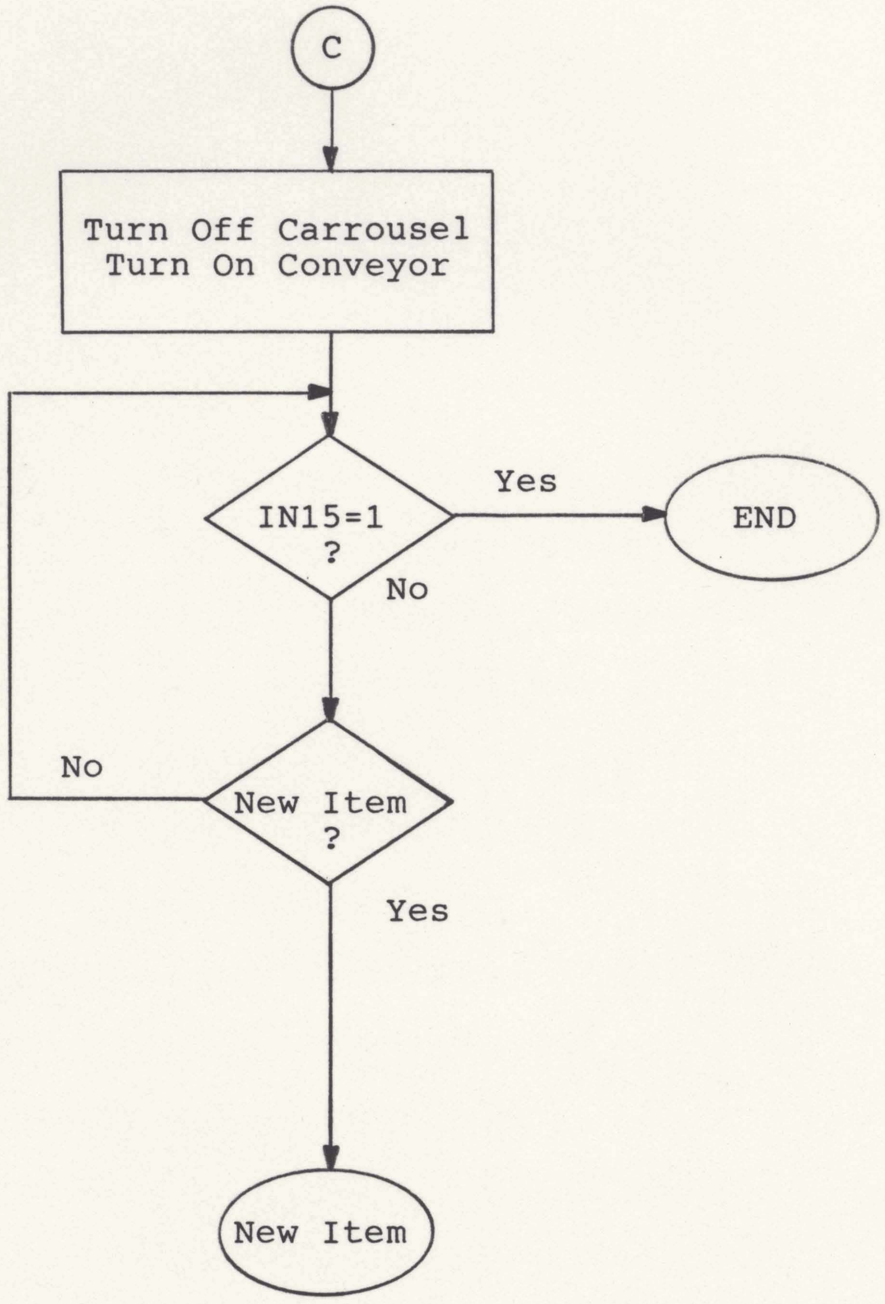
FLOW DIAGRAM FOR IPCL1.SPL




New Item
Turn Off Conveyor
Push Item Onto
Carrousel
Start Carrousel
One
Second
?
No
Yes
Carrousel
Home
?
No
Yes
Turn Off Carrousel
Turn On Drill
Turn On Down-
Drill Motor
A


A
Drill Lowered ?
No
Yes
Turn Off Down-Drill Motor
5 Seconds ?
No
Turn Off Drill Turn On Up-Drill Motor
Drill Up ?
No
Yes
B


B
Turn Off Up-Drill Motor
Turn On Conveyor
5 Seconds ?
No
New Item ?
No
Yes
Yes
New Item
Turn Off Conveyor
Turn On Carrousel
One Second ?
No
Yes
Carrousel Home ?
No
Yes
C


C
Turn Off Carrousel
Turn On Conveyor
IN15=1
?
Yes
END
No
New Item
?
No
Yes
New Item

APPENDIX C

SAMPLE PROGRAM LISTING

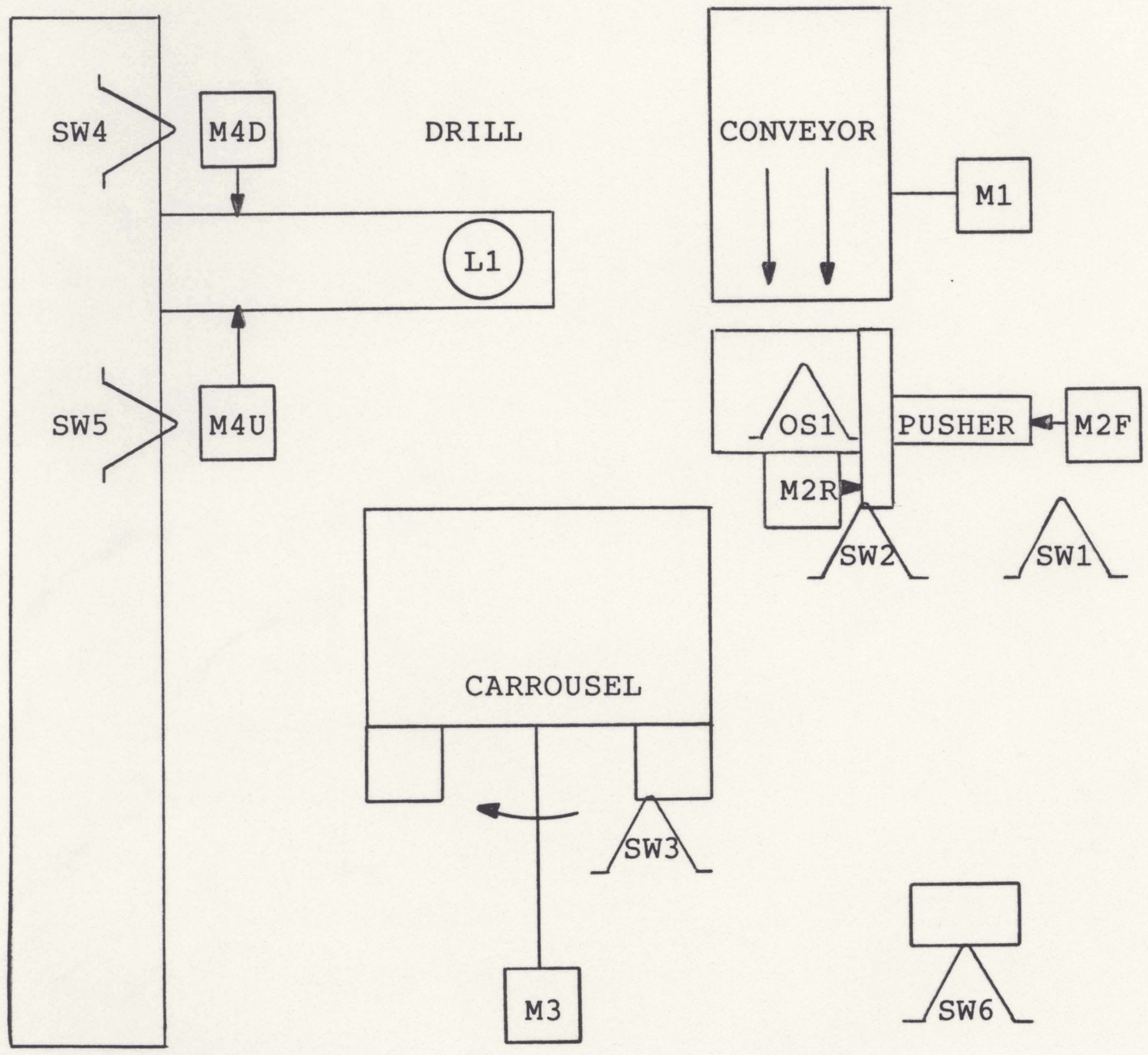
```
0010 CLR OUT ;TURN OFF ALL MOTORS & DRILL

0020 IF IN04 = 1 GOTO 0060 ;DRILL UP?

0030 SET OUT05 = 1 ;IF NOT, TURN ON UP-DRILL MOTOR

0040 IF IN04 = 0 GOTO 0040 ;NOT THERE YET

0050 SET OUT05 = 0 ;DRILL HOME, TURN OFF UP-DRILL MOTOR

0060 IF IN03 = 1 GOTO 0100 ;CARROUSEL HOME?

0070 SET OUT03=1 ;TURN ON CARROUSEL

0080 IF IN03 = 0 GOTO 0080 ;CARROUSEL NOT HOME YET

0090 SET OUT03=0 ;TURN OFF CARROUSEL, HOME

0100 IF IN01 = 1 GOTO 0140 ;PUSHER HOME?

0110 SET OUT02 = 1 ;TURN ON REVERSE PUSHER MOTOR

0120 IF IN01 = 0 GOTO 0120 ;PUSHER NOT HOME YET

0130 IF OUT02 = 0 ;TURN OFF REVERSE PUSHER MOTOR

0140 SET OUT00 = 1 ;TURN ON CONVEYOR

0150 ; START CONVEYOR

0160 IF IN00 = 1 GOTO 0160 ;WAIT FOR NEW ITEM

0170 ; NEW ITEM

0180 SET OUT00 = 0 ;TURN OFF CONVEYOR

0190 SET OUT01 = 1 ;START PUSHING ITEM ONTO CONVEYOR
```

```
0200 IF IN09 = 1 GOTO 0200 ;ITEM NOT ON CARROUSEL YET

0210 ; ITEM ON CARROUSEL

0220 SET OUT01 = 1 ;START PUSHING ITEM ONTO CONVEYOR

0230 SET OUT02 = 1 ;REVERSE PUSHER

0240 IF IN01 = 1 GOTO 0240 ;PUSHER NOT BACK YET

0250 SET OUT02 = 0 ;PUSHER BACK, TURN OFF REVERSE MOTOR

0260 ; TURN ON CARROUSEL

0270 SET OUT03 = 1 ;TURN ON CARROUSEL

0280 DELAY 1 ;WAIT ONE SECOND FOR CARROUSEL TO START

0290 IF IN03 = 1 GOTO 0290 ;WAIT FOR CARROUSEL

0300 ; STOP CARROUSEL

0310 SET OUT03 = 0 ;TURN OFF CARROUSEL

0320 ; TURN ON DRILL

0330 SET OUT06 = 1 ;TURN ON DRILL

0340 SET OUT04 = 1 ;START LOWERING DRILL

0350 IF IN05 = 1 GOTO 0350 ;DRILL NOT DOWN YET

0360 ; DRILL DOWN

0370 SET OUT04 = 0 ;TURN OFF DOWN DRILL MOTOR

0380 DELAY 5 ;DRILL FOR 5 SECONDS

0390 SET OUT06 = 0 ;TURN OFF DRILL

0400 SET OUT05 = 1 ;START RAISING DRILL

0410 IF IN04 = 1 GOTO 0410 ;DRILL NOT UP YET

0420 SET OUTP5 = 0 ;STOP UP-DRILL MOTOR

0430 ; TURN ON CONVEYOR

0440 SET OUT00 = 1 ;TURN ON CONVEYOR

0450 DELAY 5 ;WAIT 5 SECONDS

0460 IF IN00 = 1 GOTO 0170 ;NEW ITEM?
```

```
0470 ; NO NEW ITEM

0480 SET OUT00 = 0 ;TURN OFF CONVEYOR

0490 SET OUT03 = 1 ;TURN ON CARROUSEL

0500 DE 1 ;WAIT FOR CARROUSEL TO START

0510 IF IN03 = 1 GOTO 0510 ;WAIT FOR CARROUSEL

0520 SET OUT03 = 0 ;TURN OFF CARROUSEL

0530 SET OUT00 = 1 ;TURN ON CONVEYOR

0540 IF IN15 = 1 GOTO 1000 ;END SIGNAL

0550 IF IN00 = 1 GOTO 0170 ;NEW ITEM

0560 GOTO 0540 ;WAIT FOR NEW ITEM OR END SIGNAL

1000 CLR OUT ;STOP ALL MOTORS

1010 END
```

APPENDIX D

SAMPLE PROGRAM LAYOUT



MOTORS	ID	OUTXX	SWITCHES	ID	INXX
CONVEYOR	M1	OUT00	ITEM READY	OS1	IN00
FWD PUSHER	M2F	OUT01	PUSHER HOME	SW1	IN01
REV PUSHER	M2R	OUT02	PUSHER FWD	SW2	IN09
CARROUSEL	M3	OUT03	CARROUSEL HOME	SW3	IN03
DRILL DOWN	M4D	OUT04	DRILL UP(HOME)	SW4	IN04
DRILL UP	M4U	OUT05	DRILL DOWN	SW5	IN05
DRILL BIT	L1	OUT06	SHUT DOWN	SW6	IN15

APPENDIX E

IPCL1 ERROR MESSAGES

MSG2: ONLY THE FOLLOWING OPTIONS ARE ALLOWED!!!

MSG2B: L(IST), K(ILL) XXXX TO KILL STMT XXXX OR

XXXX STATEMENT TO ADD OR CHANGE STATEMENT XXXX

MSG6: NO SUCH STATEMENT NO.!

MSG7: NOT 4 DIGIT STATEMENT NO.!

MSG8: LINE TOO LONG!!

MSG9: NO PROGRAM IN BUFFER

MSGABT: PROGRAM ABORTED AT STATEMENT:

MSGLVR: VARIABLE NAME TOO LONG

MSGNOV: NO SUCH VARIABLE IN TABLE

MSGRNE: INVALID KEYWORD

LNGDIG: TOO MANY DIGITS IN VALUE

MSGNOP: NO OPERATOR IN IF STATEMENT

BIGSN: STATEMENT NO. HAS MORE THAN 4 DIGITS

NOSMNT: GOTO HAS NON-EXISTENT STATEMENT NO.

NOGOTO: NO GOTO IN IF STATEMENT

MSNOEQ: NO EQUAL SIGN IN SET STATEMENT

ILLOPR: ILLEGAL OPERATOR IN ARITHMETIC STATEMENT

SMLDIV: DIVISOR TOO SMALL

MULBIG:	MULTIPLY OVERFLOW
LNGDLY:	DELAY TIME SHOULD BE LESS THAN 100 SECONDS
NODLY:	NO DELAY VALUE IN DELAY STATEMENT
IOMSG:	I/O VARIABLE INCORRECTLY SPECIFIED
CLRION:	CANNOT CLEAR I/O VARIABLE FROM TABLE
SETINP:	CANNOT SET INPUT VARIABLES IN00-IN15

GENERAL REFERENCES

Biewald, J.; Goehner, P.; Lauber, R.; and Schelling, H. "EPOS - A Specification and Design Technique for Computer Controlled Real-Time Automation Systems," International Conference on Software Engineering, 4th, Proceedings of Technical University of Munich, Germany, September 17-19, 1979. New York: IEEE, 1979.

Bristol, E.H. "Small Languages for Large Systems and Other Uses." Proceedings of Joint Automatic Control Conference, Denver, Colorado. New York: AIChE, 1979.

Brodgesell, A., and Copeland, J.R. "CRISP: A User-Oriented Process Control Language." Proceedings of Advances in Instrumentation, Volume 28, Annual ISA Conference, 28th, Houston, Texas, October 15-18, 1973, Part 1. Pittsburgh, Pa.: ISA, 1973.

Chaky, Mike; Davidson, Glenn; Lee, Raymond; and Nichols, Jim "Processing Control BASIC Simplifies Programming." Instrument Control System 55 (January 1982): 51-54.

Copeland, J.R. "Process Control: It Isn't EPD!" ISA Transactions 15 (1976): 155 - 160.

Curtis, R.O. ISAAC/LabSoft, Boston, MA: Cyborg Corporation, 1981

Darda, L.; Gricuk, P.; and Kolodziejski, J. "Interpretive Language for Sequence Control of Chemical Processes." Software for Computer Control, Proceedings of the Second IFAC/IFIP Symposium on Software for Computer Control, Prague, Czechoslovakia, June 11-15, 1979. New York: Pergamon Press, 1979.

Feher, A.; Czeiner, N.; Csaszar, Z.; Turi, A.; Keviczky, L; Bars, R.; Haber, R.; Habermayer, M.; Hetthessy, J.; Vajk,I.; and Vajta, M. Jr. "MERCEDES - Interactive Software Package for Identification and Experimental Control of Industrial Plants by a Portable Process Computer Laboratory." _IFAC/IFIP Symposium on Software for Computer Control, 2nd, Preprint SOCOCO '79, Prague, Czechoslovakia, June 11-15, 1979._ Laxenburg, Austria: IFAC, 1979.

Fisher, D. Grant "Computer Control Offers A Future Of Changes and Challenges." _Control Engineering_ 2 (January 1982): 18-20.

Fisher, D. Grant, and Brennek, Andrew "DISCO: A Distributed, Supervisory and Control Program." _Software for Computer Control, Proceedings of the Second IFAC/IFIP Symposium on Software for Computer Control, Prague, Czechoslovakia, June 11-15, 1979._ Laxenburg, Austria: IFAC, 1979.

Freeman, L.L. "Control Applications - The Need for High Level Languages." _National Conference Publication of Instrument Engineers in Australia, Number 78/13, Conference on Microprocessor Systems, Sydney, Australia, November 21-22, 1978._ Barton, Australia: Institute of Engineers, 1979.

Gander, J.G., and Liechti, Hans U. "Real-Time Process Control Based on a High Level State Language." _Real-Time Programming 1980, Proceedings of the IFAC/IFI Workshop, Schloss Retzhof, Leibnitz, Austria, April 14-16, 1980._ Elmsford, NY: Pergamon Press, 1980.

Ghezzi, C.; Tisato, F.; and Osnaghi, A. "Language Constructs for Distributed Processing." _IFAC/IFIP Symposium on Software for Computer Control, 2nd, Preprint SOCOCO '79, Prague, Czechoslovakia, June 11-15, 1979._ Laxenburg, Austria: IFAC, 1979.

Guillespie, D.M. "Hierarchical Languages for Process Control," _Proceedings of the Joint Automatic Control Conference, San Francisco, California, June 22-24, 1977_ . New York: IEEE, 1979.

Harrison, Thomas J. (Editor). Distributed Computer Control Systems, Proceedings of the IFAC Workshop, 1979, Tampa, Florida, October 2-4, 1979. Elmsford, NY: Pergamon Press, 1980.

Hetthessy, J; Nagy, D.; and Zarandi, E. "MICROCOUNT Interpretive Control Language Based on Intel 8080." IFAC/IFIP Symposium on Software for Computer Control, 2nd, Preprint SOCOCO '79, Prague, Czechoslovakia, June 11-15, 1979. Laxenburg, Austria: IFAC, 1979.

Krull, Fred N. "Experience with ILIAD: A High-Level Process Control Language." Communications of the ACM 24 (February 1981): 66-72.

Lewis, A., and Trainito, G. "Implementation of a Standard Language for Real-Time Distributed Process Control." Software for Computer Control, Proceedings of the Second IFAC/IFIP Symposium on Software for Computer Control, Prague, Czechoslovakia, June 11-15 1979. New York: Pergamon Press, 1979.

Ludewig, Jochen "Process Control Software Specification in PCSL." Real Time Programming 1980, Proceedings of the IFAC/IFIP Workshop, Schloss Retzhof, Leibnitz, Austria, April 14-16, 1980. Elmsford, NY: Pergamon Press, 1980.

Moore, C.H., and Rather, E.D. "Use of 'FORTH' in Process Control." International Microcomputer/Minicomputer/ Microprocessor '77, Proceedings of an International Conference, Geneva, Switzerland, May 24-26, 1977. Guildford, Surrey, England: IPC Science and Technology Press, 1977.

Novak, M. (Editor). Software for Computer Control, Proceedings of the Second IFAC/IFIP Symposium on Software for Computer Control, Prague, Czechoslovakia, June 11-15, 1979. New York: Pergamon Press, 1979.

Pageler, Evan I. "Interactive Process Control Language Designed for the Control Engineer." *Advances in Instrumentation, Volume 34, Part 2, 1979, Proceedings of the ISA Conference and Exhibition, Chicago, Illinois, October 22-25, 1979.* Pittsburgh, Pa.: ISA, 1979.

PROSPRO II (TSX/1800) PROcess Systems PROgrams. White Plains, NY: International Business Machine Corporation, 1970.

Shaw, I.L.; Edbald, W.A.; and Pavlovic, A.M. "Process Control Languages - Designer's Perspective of Adequacy and Future Requirements." *Proceedings of COMPSAC '78: IEEE Computer Society International Software and Applications Conference, 2nd, Chicago, Illinois, November 13-16, 1978. New York: IEEE,* 1978.

Strange, Charles Clinton. "DIPSTICK - A PROCESS CONTROL LANGUAGE." Masters Research Report, University of Central Florida, Orlando, 1985.

VAL Primer 398H3A. An Introduction to Basic Programming of PUMA Robot using VAL Language. Danbury, CT: Unimation Inc., 1980.

Walter, C. "Structuring Language for Computer Controlled Multilevel Systems." *Real Time Programming 1980, Proceedings of the IFAC/IFIP Workshop, Schloss Retzhoff, Leibnitz, Austria, April 14-16, 1980.* Elmsford, NY: Pergamon Press, 1980.

Windal, G. "Universal Software Interfaces for Distributed Process Control by Micro-Computers." *Software for Computer Control, Proceedings of the Second IFAC/IFIP Symposium on Software for Computer Control, Prague, Czechoslovakia, June 11-15, 1979.* New York: Pergamon Press, 1979.