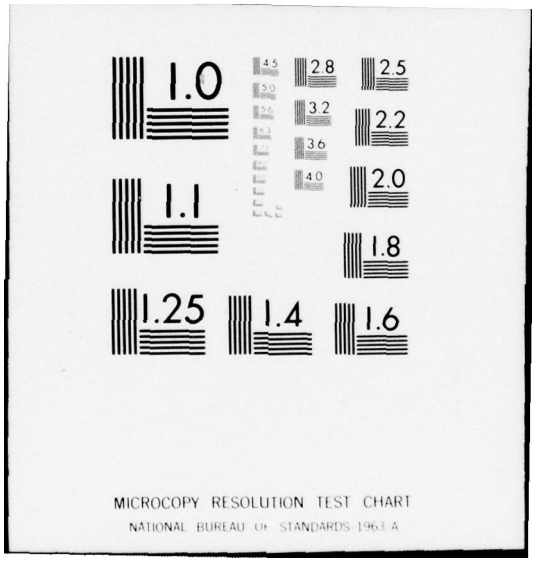
AD-A038 058 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIO--ETC(U)
JAN 77 N00014-76-C-0732

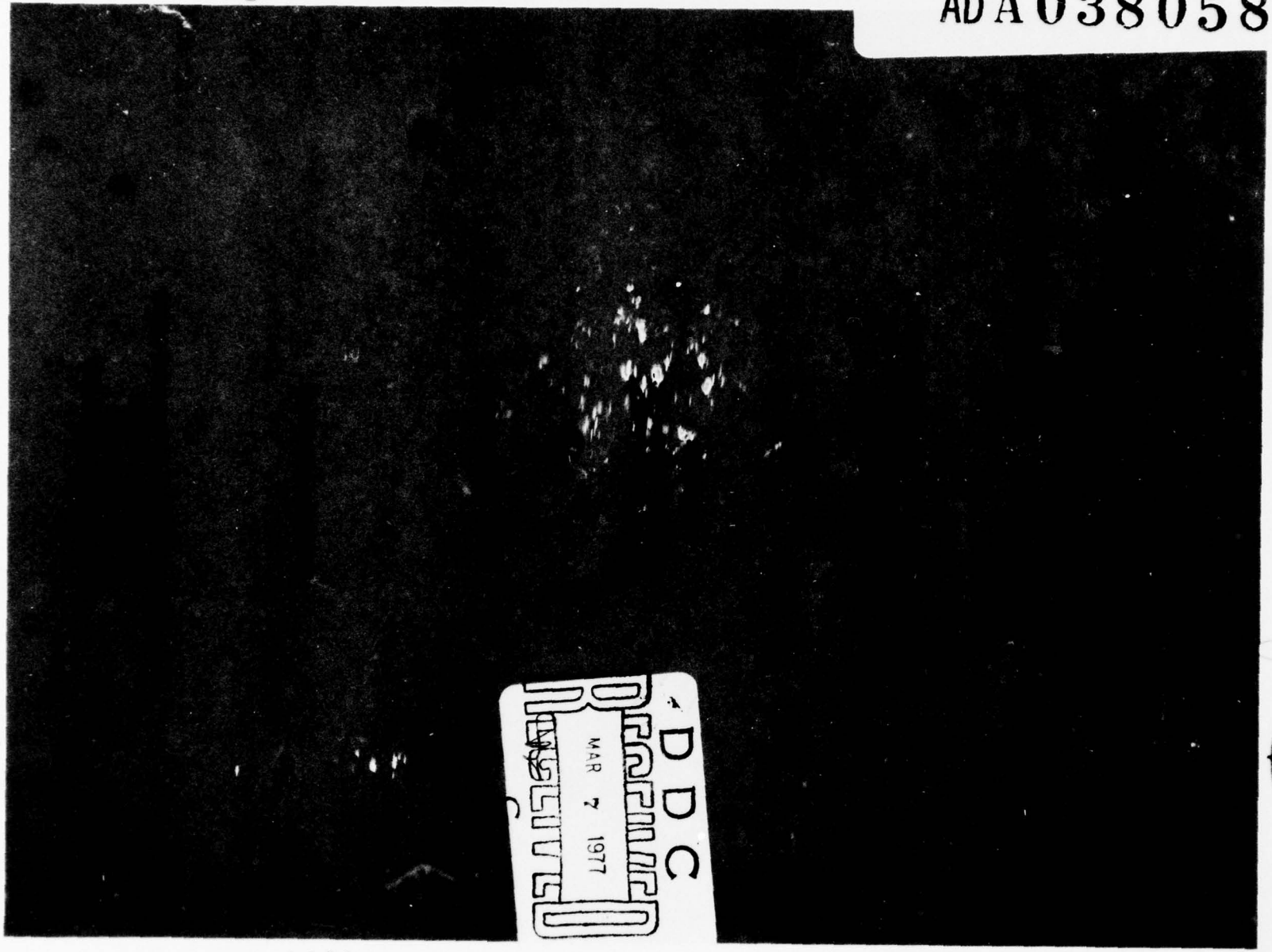
UNCLASSIFIED NL

1 OF 4
ADA038058


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

DDC FILE COPY

AD A038058



DDC
RECEIVED
MAR 7 1977

BEST AVAILABLE COPY

This Report is Published as Part of
Engineering Experiment Station Bulletin 143 Series

Schools of Engineering
Purdue University
West Lafayette, Indiana 47907

BEST AVAILABLE COPY

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: NR049-388	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER: Final rept. 1 Feb 76 - 31 Jan 77
4. TITLE *(and Subtitle)*: SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIONAL PURDUE WORKSHOP ON INDUSTRIAL COMPUTER SYSTEMS. PART IV. SOME REPORTS ON THE STATE-OF-THE-ART AND FUNCTIONAL REQUIREMENTS FOR FUTURE APPLICATIONS.	5. TYPE OF REPORT & PERIOD COVERED: Final 2/1/76 - 1/31/77
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0732
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Purdue Laboratory for Applied Industrial Control, Schools of Engineering, Purdue University, West Lafayette, Indiana 47907	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Department of the Navy, Office of Naval Research, Arlington, Virginia 22217	12. REPORT DATE: January 1977
	13. NUMBER OF PAGES: 320 + xvi
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: 320p.
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

DDC MAR 1977 B

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Approved for public release; distribution unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This volume represents Part ~~IV~~ 4 of a six volume set reproducing the major work accomplished by the International Purdue Workshop on Industrial Computer Systems during the past eight years. This material is reprinted from the Minutes of the several individual meetings of the Workshop and represents the work carried out by the standing committees of the Workshop.

408244

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIONAL PURDUE WORKSHOP ON INDUSTRIAL COMPUTER SYSTEMS

PART IV

SOME REPORTS ON THE STATE-OF-THE-ART AND FUNCTIONAL REQUIREMENTS FOR FUTURE APPLICATIONS

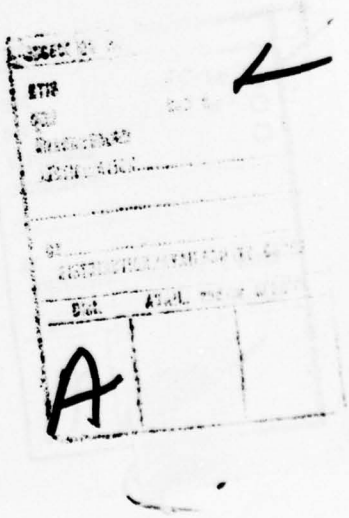
Prepared for

Department of the Navy

Office of Naval Research

January 1977

Distribution is Unlimited



Purdue Laboratory for Applied Industrial Control
Schools of Engineering
Purdue University
West Lafayette, Indiana 47907

FOREWORD

This material is published as part of Contract N00014-76-C-0732 with the Office of Naval Research, United States Department of the Navy, entitled, The International Purdue Workshop on Industrial Computer Systems and Its Work In Promoting Computer Control Guidelines and Standards. This contract provides for an indexing and editing of the results of the Workshop Meetings, particularly the Minutes, to make their contents more readily available to potential users. We are grateful to the United States Navy for their great help to this Workshop in this regard.

Theodore J. Williams

PRECEDING PAGE BLANK-NOT FILMED

TABLE OF CONTENTS

		Page
Report Documentation Page		i
Foreword		v
List of Figures		ix
List of Tables		xi
Background Information on the Workshop		xiii
INTRODUCTION		1
SECTION I	- FUNCTIONAL REQUIREMENTS FOR INDUSTRIAL COMPUTER SYSTEMS	9
SECTION II	- FUNCTIONAL REQUIREMENTS FOR LANGUAGE FEATURES FOR A PROCEDURAL LANGUAGE FOR INDUSTRIAL COMPUTERS	59
SECTION III	- REQUIREMENTS FOR DATA COLLECTION COMPUTERIZED SYSTEM CHECK-OUT AND SEQUENCING CONTROL FUNCTIONS	87
SECTION IV	- FUNCTIONAL NEEDS FOR REAL-TIME ENHANCEMENTS TO MINIMAL BASIC	97
SECTION V	- FUNCTIONAL GUIDELINES FOR INDUSTRIAL CONTROL COMPUTER SYSTEMS	109
SECTION VI	- STATE-OF-THE-ART OF PROCEDURAL PROGRAMMING LANGUAGES FOR INDUSTRIAL APPLICATIONS	251
SECTION VII	- SOME ADDITIONAL IMPORTANT TOPICS PRESENTED TO THE WORKSHOP REGARDING THE STATE-OF-THE-ART OF APPLICATIONS	281

PRECEDING PAGE BLANK-NOT FILMED

TABLE OF CONTENTS (CONT.)

Page

Report on Japanese Industrial Computer System Questionnaire Survey 283

Process Computer Applications in the Federal Republic of Germany Past, Present and Future of Funded Activities. 301

Pertinent Time Constants of Processes and Management Functions. 315

LIST OF FIGURES

Figure numbers used here are the same as those assigned to these figures in their previous publication in the Minutes of the International Purdue Workshop on Industrial Computer Systems. Therefore they will not be in strict numerical sequence here.

Page

SECTION II - FUNCTIONAL REQUIREMENTS FOR LANGUAGE FEATURES FOR A PROCEDURAL LANGUAGE FOR INDUSTRIAL COMPUTERS

FIGURE 1 - Task State Diagram with Some Possible Transitions. . . 65

FIGURE 2 - State Diagram of Proposed Tasking System 74

SECTION V - FUNCTIONAL GUIDELINES FOR INDUSTRIAL CONTROL COMPUTER SYSTEMS

FIGURE 1 - Outline of Hierarchy Organzation for Standardized Process Computer Control System. . . . 115

FIGURE 2 - Concept of Remote Multiplexer and Data Highway System for Process Data Acquisition . . . 121

FIGURE 3 - Diagram of the Polling Sequence Used for all Point Sampling for Control of Large Industrial Complex. 124

FIGURE 4 - Some Features of Operator's Consoles for the Proposed Process Control System. . . . 127

FIGURE 5 - Block Diagram of Modular Industrial Control Program System. 134

LIST OF FIGURES (CONT.)

Page

FIGURE 6 - Diagram of Relationship of Machine Hardware Functions, Vendor Supplied Programming and User Developed Higher Level Language Programs. . . 135

FIGURE 7 - Computer-Instrumentation Coordination Responsibility. 157

FIGURE 8 - Stem Force Versus Supply Pressure for Series 470 Actuators Equipped with Type 3570 Positioner. 214

FIGURE 9 - Frequency Response Curves for Type 470 Actuators . . . 214

FIGURE 12 - Relationship of Mean Time Between Failures and Allowable Repair Time for Direct Digital Control Computers. 227

FIGURE 13 - Some Examples of Control Loops. 236

FIGURE 14 - Relationship of Allowed Computer Downtime per Failure to Type of Backup Used . . . 247

SECTION VII - SOME ADDITIONAL IMPORTANT TOPICS PRESENTED TO THE WORKSHOP REGARDING THE STATE-OF-THE-ART OF APPLICATIONS

FIGURE 1 -- Process Time Constants, Seconds. 319

FIGURE 2 - The Dynamics of Human & Machine Organizations in Manufacturing, Planning & Control. 320

LIST OF TABLES

Table numbers used here are the same as those assigned to these tables in the previous publications of these documents in the Minutes of the International Purdue Workshop on Industrial Computer Systems. Therefore they will not be in strict numbrical succession here.

Page

INTRODUCTION

TABLE I - A List of All Documents Produced in This Summary of the Work of the International Purdue Workshop on Industrial Computer Systems. 3

TABLE II - List of Tutorials Presented at Purdue Workshop Meetings . . . 5

SECTION II - FUNCTIONAL REQUIREMENTS FOR LANGUAGE FEATURES FOR A PROCEDURAL LANGUAGE FOR INDUSTRIAL COMPUTERS

TABLE I - Summary of Tasking Functions

SECTION V - FUNCTIONAL GUIDELINES FOR INDUSTRIAL CONTROL COMPUTER SYSTEMS

TABLE I - Assignments of Tasks within the Control Hierarchy as it Affects the Standard Process Control Computer System 116

TABLE II - Minimum Required Characteristics and Capabilities of Computer Mainframes Used in the Proposed Standard Process Control Computer System. 117

TABLE III - Some Provisions of System (Non-Computer Hardware Used for Standard Process Control Computer System for Industrial Complexes. . . 118

LIST OF TABLES (CONT.)

Page

TABLE IV - Requirements on Operation of Production Control Computer Systems for an Industrial Complex. 122

TABLE V - Data Load per Production Control Computer System (Level 3) for an Industrial Complex . . 128

TABLE VI - Programming Requirements for Achieving a Desirable Standard Process Computer Control System. 131

TABLE VII - Stroking Speed in Seconds for Control Valves 213

TABLE VIII- Piston Actuators 214

TABLE IX - Series 3560 V/P Valve Positioner 215

TABLE X - Distribution of Valves According to Line Size Petrochemical Plant Applications 217

SECTION VI - STATE-OF-THE-ART OF PROCEDURAL PROGRAMMING LANGUAGES FOR INDUSTRIAL APPLICATIONS

TABLE 2.1 - Comparison of FORTRAN Extensions 263

BACKGROUND INFORMATION ON THE WORKSHOP

The International Purdue Workshop on Industrial Computer Systems, in its present format, came about as the results of a merger in 1973 of the Instrument Society of America (ISA) Computer Control Workshop with the former Purdue Workshop on the Standardization of Industrial Computer Languages, also cosponsored by the ISA. This merger brought together the former workshops' separate emphases on hardware and software into a stronger emphasis on engineering methods for computer projects. Applications interest remains in the use of digital computers to aid in the operation of industrial processes of all types.

The ISA Computer Control Workshop had itself been a renaming in 1967 of the former Users Workshop on Direct Digital Computer Control, established in 1963 under Instrument Society of America sponsorship. This Workshop in its annual meetings had been responsible for much of the early coordination work in the field of direct digital control and its application to industrial process control. The Purdue Workshop on Standardization of Industrial Computer Languages had been established in 1969 on a semiannual meeting basis to satisfy a widespread desire and need expressed at that time for development of standards for languages in the industrial computer control area.

The new combined international workshop provides a forum for the exchange of experiences and for the development of guidelines and proposed standards throughout the world.

Regional meetings are held each spring in Europe, North America and Japan, with a combined international meeting each fall at Purdue University. The regional groups are divided into several technical committees to assemble implementation guidelines and standards proposals on specialized hardware and software topics of common interest. Attendees represent many industries, both users and vendors of industrial computer systems and components, universities and research institutions, with a wide range of experience in the industrial application of digital systems. Each workshop meeting features tutorial presentations on systems engineering topics by recognized leaders in the field. Results of the workshop are published in the Minutes of each meeting, in technical papers and trade magazine articles by workshop participants, or as more formal books and proposed standards. Formal standardization is accomplished through recognized standards-issuing organizations such as the ISA, trade associations, and national standards bodies.

The International Purdue Workshop on Industrial Computer Systems is jointly sponsored by the Automatic Control Systems Division, the Chemical and Petroleum Industries Division, and the Data Handling and Computations Division of the Instrument Society of America, and by the International Federation for Information Processing as Working Group 5.4 of Technical Committee TC-5.

The Workshop is affiliated with the Institute of Electrical and Electronic Engineering through the Data Acquisition and Control Committee of the Computer Society and the Industrial Control Committee of the Industrial Applications Society, as well as the International Federation of Automatic Control through its Computer Committee.

INTRODUCTION

The Office of Naval Research of the Department of the Navy has made possible an extensive report, summary and indexing of the work of the International Purdue Workshop on Industrial Computer Systems as carried out over the past eight years. This work has involved twenty-five separate workshop meetings plus a very large number (over 100) of separate meetings of the committees of the workshop and of its regional branches. This work has produced a mass of documentation which has been severally edited for the original minutes themselves and then again for these summary collections.

A listing of all of the documentations developed as a result of the U.S. Navy sponsored project is given in Table I at the end of this introduction. The workshop participants are hopeful that it will be helpful to others as well as themselves in the very important work of developing guidelines and standards for the field of industrial computer systems in their many applications.

A major part of the efforts of the Workshop at its several meetings has been a continuing review of the state-of the-art and the needs of the computer field in general and of the industrial computer applications field in particular. This has been accomplished through a set of tutorial presentations made to the workshop attendees by recognized experts in the field and through specific studies undertaken by the several

committees of the Workshop. The tutorials presented to date are listed in Table II. None are reproduced here because of the volume involved. Those interested are referred to the original Minutes as noted. This Part of the Summary of the results of the Workshop is devoted to reports of the second kind, i.e., those developed by the several committees of the Workshop is part of their regular studies. A large part of the specific reports generated by the committees were those concerning the functional requirements for the several industrial computer application areas. Several of these are included in this Part.

TABLE I

A LIST OF ALL DOCUMENTS PRODUCED IN THIS SUMMARY OF THE WORK OF THE INTERNATIONAL PURDUE WORKSHOP ON INDUSTRIAL COMPUTER SYSTEMS

1. The International Purdue Workshop on Industrial Computer Systems and Its Work in Promoting Computer Control Guidelines and Standards, Report Number 77, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, Indiana, Originally Published May 1976, Revised November 1976.

2. An Index to the Minutes of the International Purdue Workshop on Industrial Computer Systems and Its Predecessor Workshops, Report Number 88, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, Indiana, January 1977.

3. A Language Comparison Developed by The Long Term Procedural Languages Committee - Europe, Committee TC-3 of Purdue Europe, Originally Published January 1976, Republished October 1976.

4-9. Significant Accomplishments and Documentation of the International Purdue Workshop on Industrial Computer Systems.

Part I - Extended FORTRAN for Industrial Real-Time Applications and Studies in Problem Oriented Languages.

Part II - The Long Term Procedural Languages

PART III - Developments in Interfaces and Data Transmission, in Man-Machine Communications and in the Safety and Security of Industrial Computer Systems.

Part IV - Some Reports on the State-of-the-Art and Functional Requirements for Future Applications.

Part V - Documents on Existing and Presently Proposed Languages Related to the Studies of the Workshop.

TABLE I (Cont.)

PART VI - Guidelines for the Design of Man/Machine Interfaces for Process Control

All dated January 1977

The latter seven documents are also published by the Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, Indiana.

TABLE II

LIST OF TUTORIALS PRESENTED AT

PURDUE WORKSHOP MEETINGS

I. International Meeting - Fall 1973

1. Mr. Alonzo R. Parsons, Honeywell, Inc.

"National and International Standards Organizations and Their Relation to the Work of the Purdue Workshop"
November 26, 1973

2. Mr. Donald C. Loughry, Hewlett-Packard Co.

"Byte Serial - Bit Parallel Dataway Standards Proposals of Working Group 3 - IEC TC - 65"
November 27, 1973

Both tutorials were published in the Minutes, First International Meeting, November 1973.

II. American Regional Meeting - Spring 1974

3. Professor Peter Denning, Purdue University

"On the Significance of Virtual Storage"
April 1, 1974

4. Mr. David Frost, Honeywell, Inc.

"An Overview of Structured Programming"
April 1, 1974

5. Mr. Frank Engel, Jr., ANSI X3J3

"The Work of the ANSI FORTRAN Committee X3J3 and Its Relation to the Purdue Workshops"
April 2, 1974

All of the above tutorials were published in Minutes of Spring Meetings 1974.

TABLE II (Cont.)

III. International Meeting - Fall 1974

6. Mr. Bobby L. Hartway, Los Alamos Scientific Laboratory

"A Tutorial on Human Factors"
October 8, 1974

7. Mr. Robert Van Naarden, Digital Equipment Corporation

"A Tutorial on Micro-Computers"
October 8, 1974

Tutorials listed above were published in the Minutes, Second Annual Meeting, October 1974.

IV. American Regional Meeting - Spring 1975

8. Mr. Bobby L. Hartway, Los Alamos Scientific Laboratory

"Human Factors in Engineering"
April 15, 1975

9. Mr. William B. Field, Union Carbide Corporation

Mr. F. Mark Rehnvorg, ACCO-Bristol Division

"Implementation of a Problem Oriented Language Using a Microprocessor"
April 15, 1975

Tutorials listed above were both published in the Minutes, Spring 1975.

Far East Regional Meeting - Spring 1975

10. Mr. S. Gocho, Nippon Mining Company

"On the Industrial Computer Networks"
July 2, 1975

This tutorial was not available for publication.

TABLE II (Cont.)

V. International Meeting - Fall 1975

11. Mr. William B. Field, Union Carbide Corporation

"A Tutorial on Distributed Control via Dynamic Simulation"
October 28, 1975

12. Mr. William B. Field, Union Carbide Corporation

"A Tutorial on Systems Reliability and Safety"
October 30, 1975

Copies of above listed tutorials were not available.

VI. American Regional Meeting - Spring 1976

13. Mr. Fred G. Sheane, Chairperson, Imperial Oil Enterprises, Ltd.

Mr. John Lee, Foxboro Company

Mr. C. R. Stewart, Honeywell, Inc.

Mr. Vincent J. McKnight, Taylor Instrument Companies

Mr. Bruce Allen, ACCO Corp.

"Impact of Microprocessor Based Digital Instrumentation"
April 6, 1976

14. Ms. Mary Adix, General Motors Corporation

"ILIAD: A High-Level Language for Industrial Control Applications"
April 7, 1976

15. Mr. Warren E. Loper, Naval Electronics Development Center

"The TINMAN Requirements for the DOD Higher Level Common Language"
April 6, 1976

TABLE II (Cont.)

16. Dr. James S. Miller, Intermetrics, Inc.

"The CS-4 Language"
April 7, 1976

Copies of these discussions were published in the Minutes of the American Regional Meeting, April 1976.

SECTION I

FUNCTIONAL REQUIREMENTS FOR INDUSTRIAL COMPUTER SYSTEMS

One of the early committees of the Purdue Workshop on Standardization of Industrial Computer Languages was one called the Functional Requirements Committee. Its job was to develop those functional requirements of an industrial computer system which would serve as a basis for the efforts of the Workshop in its developing of standard industrial computer languages. This committee finished its work at the time of the Fifth Meeting of the Workshop and was then discharged. The attached report is the findings of the committee which were published on pp. 53-99 of the Minutes of the Fifth Workshop.

FUNCTIONAL REQUIREMENTS FOR INDUSTRIAL COMPUTER SYSTEMS

(Adopted by the Workshop on Standardization of Industrial Computer Languages - May 6, 1971)

FUNCTIONAL REQUIREMENTS COMMITTEE OBJECTIVE

The main objective of this committee has been to develop functional requirements for industrial computer systems to serve as a basis for the development of standard industrial computer programming languages. These functional requirements should be sufficiently broad to encompass most applications of industrial computer systems, but it is not required that all functional requirements apply to all application areas or any specific installation. If a function is to be implemented, it should satisfy the applicable functional requirements, as defined herein.

The intent is that the following functions will be provided by some means in the languages and/or standards being developed by the various subcommittees of this Workshop. Various examples given are for illustrative purposes only; it is not intended that the range of application of these standardized functions be particularly limited to the range of the examples used.

PRECEDING PAGE BLANK-NOT FILMED

SUMMARY OF RECOMMENDED FUNCTIONAL REQUIREMENTS FOR INDUSTRIAL COMPUTER SYSTEMS

I. INTRODUCTION

Although industrial computers are used for a wide variety of applications, certain similarities exist in such a way that standardization can offer an economic benefit, through reduced programming effort, for users and suppliers of these systems.

An industrial computer system is one that communicates with a manufacturing process in what is called "real time", getting information from the process at a rate high enough to be of use in some form of control of the process itself. This includes a wide dynamic range of applications, from servo systems with frequency responses of many cycles per second, into the management information and control area dealing with business cycles of several years per cycle. The common feature, communication with the process, is worthy of standardization.

All processes are managed by people, so all computer systems have some means for man/machine communications, and these means are worthy of standardization to some extent, even though the hardware used and the man in the loop in particular situations may vary considerably.

Various common functions are not all equally suited to standardization. Those functions that are more suited to

standardization are also more suited to use of "problem-oriented" languages, and can be handled as function modules, or packages. Other, less disciplined functions may require generation of application-oriented programming packages, using "procedural" languages, but the procedural languages may be worthy of standardization even though the packages generated bear no similarity whatsoever. Thus, standardization is directed toward that effort where it can provide a benefit, not a penalty.

The organization of functional requirements that follows has no significance. It was used by the Functional Requirements Committee of the Workshop as a way to organize its work, so numerous cross-references will be found in the result. The whole task could just as well have been divided into "System Generation" and "System Maintenance", for example. Current state-of-the-art limitations have been ignored, and the functional requirements should be interpreted with this in mind. (For example, consider the advent of digital transducers. Functionally, they are the same as the familiar analog inputs, so no distinction is made here. Even logical, event-oriented inputs may be considered to be one-bit analog-to-digital conversions of a process measurement.)

No hardware configuration limitations have been considered. It may well be the case that a standardized scheme for system generation requires more hardware, temporarily, than can

be justified for the industrial application. Rather than exclude small systems from the benefits of standardization in system generation, or require all systems to be larger than the run-time application requires, the standards do not require system generation to be done on the object machine, nor do they prevent it from being done on the object machine. In fact, time-sharing service vendors may well become the vehicle for implementation of the compilers and utilities with all the features of these standards.

Note that many of the features mentioned may not have any implication in the form a language takes and that the effects (where present) may differ in the problem oriented and procedural languages. It is felt that for completeness, a statement of functional requirements should be all-encompassing. It may well be that the adoption of these functional requirements as a basic standard results in the implementation of some functions as hardware rather than software. That this should occur is just as valid an objective as that a standard language should emerge. One example where the functional requirements do extend considerably beyond the requirements for language definition is the section on system maintenance. The justification is as stated above.

II. PROCESS I/O

A. Process Measurement Inputs

1. Scan:

a. Scheduled: Operation allows the specification of frequency, grouping and phasing for each input.

It should be possible to specify a scan frequency or processing interval for each input. Completely random specification of frequency will not be necessary if enough different time classes are included.

It should be possible to group any given input with any other input or inputs so that they are scanned consecutively.

It should be possible to distribute the scanning of the inputs in a given time class over the period of the time class so that the scanner is not overloaded at certain times.

For example, if phasing were not done on a system containing 0.25, 1, 5, 15, 30, and 60-second time classes, they would all come due every minute, possibly causing the scanner to fall behind. Phasing, however, must not interfere with the above grouping requirement.

b. Demand: It should be possible for interrupt routines or other programs to make demand scan requests for individual or multiple readings of any points. These requests should be done as soon as possible after they are requested, and may optionally take precedence over regular cyclical scan points. Conversion to engineering units should be optional.

2. Validation: The input value of each scanned input should be checked for validity limits and/or illegal configuration.

It should be possible to remove an input from scan, or limit checking when it has failed validity limits for n consecutive times, where n is a system generation parameter.

It should be possible to output a message with the out-of-limit value when a point has failed validity limits n consective times.

3. Conversion:

a. Compensation: input readings should be compensated for zero offset before applying conversion equations.

b. Standard equations: Provision should be made for several different conversion equations. For example

(1) Thermocouples including the following:

Iron-Constantan

Copper-Constantan

Chromel-Constantan

Chromel-Alumel

Platinum - Platinum Rhodium (10%)

Platinum - Platinum Rhodium (13%)

Conversion co-efficients should be supplied which will provide errors within the limits of ISA standard C96.1. Input readings should be compensated for thermocouple reference before conversion.

(2) Linear conversion of the form MX + B

(3) Polynomial conversion

(4) Flow conversion incorporating a square-root function

a. Liquid Flow:

Uncompensated

Temperature Compensated

Gravity Compensated

Temperature and Gravity Compensated

b. Gas and Steam Flow

Uncompensated

Pressure compensated

Temperature Compensated

Gravity Compensated

Combination of above

(5) No conversion -- readings stored as read

(6) Special equations: user should have the ability to specify any conversion, such as table look-up for code conversions, and to relate scan frequency to filter and/or conversion processing interval.

4. Filtering: Each input should have the option of having its value filtered using a digital filter factor specified on a point-by-point basis. The user should have the ability to specify any filter equations, and to relate scan frequency to filter processing interval.

5. Alarming:

a. Each input should have the option of having its value checked against both high and low limits and a rate of change limit. When limit violations occur it should be possible to:

(1) pring a message

(2) cause the execution of a program

(3) open or close a contact (user should have

the ability to specify the list of contacts).

(4) any combination of the above

B. Process Outputs

1. Two common internal forms of outputs are absolute and incremental values. Both forms should be supported.

2. Output Generation: The generation of outputs should allow for demand outputs or the specification of frequency and phasing of each output.

3. Conversion from engineering units to count value

a. Type of Conversion

Conversion equations should include at least:

Linear

Quadratic

Code Conversions

b. Limit Checking

High and low limits should be checked for each output calculated. Violation may optionally provide linkage to a specified program for appropriate action.

c. Deviation Limits

Deviations are defined as the difference between the current output value and the newly calculated output value. Maximum absolute deviations may optionally be specified for each output which

are the limiting values of any change in an output.

If the deviation limit is exceeded, a program linkage to a specified program can be established. Dead-band capability should be provided.

C. Event-Oriented Input

Interrupts should be caused by changes in state of single bit inputs from one logical condition to the other, but not the reverse.

III. MAN/COMPUTER COMMUNICATIONS

A. General

1. All operator I/O requests should be acknowledged within one second. A response should be given to the requester to indicate either a valid or invalid request was made. If an invalid request has been made, the system should indicate the reason for trouble.
2. Provision should be made to prevent the process operator from making engineering and programming date entries

B. Process Operator and/or Supervision

1. Active entry functions:

   These operations should be available to change the operating status of the system. The implementation of the operator entry functions should be done in such a way as to require his performing certain additional coded (keyed) actions so as to minimize the possibility of inadvertent entry of erroneous information. All parameters available to the operator should have a range of acceptable values and the operator should be notified when his entry is not within the acceptable range.

   a. Process I/O scan--the operator should be able to activate any process I/O scan point.

b. Alarm Scan: The operator should be allowed to:

(1) Activate or deactivate any alarm.

(2) Change the high and low alarm limits.

(3) Change the rate-of-change limits.

c. Logging: The operator should be allowed to:

(1) Add/delete a variable to/from a log.

(2) Request logs on a timed frequency or on demand. This includes designation of an interval.

d. Trend recording: The operator should be allowed to specify the pen number and variable along with the desired frequency, the desired range and intercept.

e. Control: Those parameters which require knowledge of control theory and timing should not be considered as operator functions. He should be allowed to:

(1) Change the set point of a loop.

(2) Activate or deactivate a loop.

f. Demand Functions: Certain functions, including user application programs, should be made available to the operator on demand. He should have the capability to:

(1) Select the functions

(2) Specify a specific time for execution or

(3) Specify a time interval for execution or,

(4) Demand immediate execution.

(5) Terminate the function.

g. Miscellaneous: The operator should be able to:

(1) Set the time and date.

(2) Activate or deactivate selected peripheral I/O devices.

(3) Enter pre-specified process data.

2. Passive Entry Functions:

These operations should be made available to interrogate the operating status of the system.

a. Process I/O scan: The operator should be able to list or display:

(1) The status of a variable (active/inactive)

(2) The value of a variable or of a raw data input.

(3) The identification of all variables deactivated from the scan.

b. Alarm scan: The operator should be permitted to:

(1) List or display the alarm status of any alarmed variable.

(2) List the identification along with limits and value of those variables currently in alarm.

c. Logging: The operator should be allowed to:

(1) Obtain a list of those logs currently inactive.

(2) Obtain a list of those logs currently on automatic request along with the interval for each.

d. Trend recording: The operator should be able to obtain the current status of each trend pen including the identification and present value of the variable along with its range, its intercept and its scan frequency.

e. Control: The operator should be able to list or display:

(1) The current set point of a loop,

(2) Whether a loop is active or inactive.

(3) The control limits.

f. Demand functions: He should be allowed to list a menu of the demand functions, their parameters with regard to demanding a program, and status (active/inactive).

g. Miscellaneous--he should be able to demand the current:

(1) Time and date.

(2) Status of selected I/O peripherals.

(3) Value of preselected process data.

C. Control Engineer

1. Active Entry Functions

a. Process output: The control engineer should be able to activate or deactivate and control any digital/analog device. However, these actions should not be permitted unless the control loop corresponding to the device is deactivated.

b. Control: The engineer should be able to change all parameters having to do with a control loop, such as algorithm constants, type of algorithm to be used, maximum allowable change, control frequency, cascading optings, etc.

c. Process I/O scan: The engineer should be able to change other variable parameters, such as as scan frequency, conversion coefficients, validation limits, conversion type, filter constants, engineering units, flow coefficients, type compensation, and others as necessary.

d. Data age checking: He should be allowed to:

(1) Specify any variable for age checking as a validity check.

(2) Specify the age limit for any variable being checked.

2. Passive Entry Functions

a. Digital I/O: The engineer should be able to

obtain a print-out or display of the configuration of a specified group or bit (s).

b. Control: He should be able to obtain a list of all parameters and their values, having to do with a specified control loop.

c. Process I/O scan: The engineer should be able to list and/or display the value of any variable parameter, such as conversion coefficents, validation limits, conversion type, filter constants engineering units, flow coefficients, type of compensation, point scan frequency, and others.

D. Programmer

1. Active Entry Functions

a. The programmer should be permitted to change the contents of from 1 to n consecutive locations in core or bulk storage through the programmer's I/O device, in decimal and in the number system used by the system assembler. Any changes to either core or bulk storage should be accompanied by a print-out of the contents of the altered location(s) before and after the change.

b. The programmer should be able to cause the transfer of from 1 to n locations to/from core

or bulk storage.

c. He should be able to read (card or tape) and store 1 to n locations into core or bulk storage.

2. Passive Entry Functions

a. The programmer should be able to list the contents of from 1 to n consecutive core or bulk storage locations on the programmer's I/O device in decimal and in the number system used by the system assembler.

b. He should be able to punch (on cards or tape) from 1 to n consecutive locations of core or bulk storage.

3. Program Loading and Linking: The programmer should be able to load (via cards or tape) and link programs to the operating system on-line, and enter the parameters necessary to make the program function in the system.

IV. OPERATING SYSTEMS

A. Interrupt Handling

1. Definitions: For the purposes of this specification, "interrupt" shall refer to suspension of the execution of a routine as a result of a hardware or program generated signal. Interrupt handling is the procedure which the operating system follows when responding to an interrupt.

2. Interrupt levels: In cases where the hardware configuration has a number of interrupts, the operating system should be capable of responding to these according to the assigned level or priority of each interrupt.

3. Interrupt conditions: With respect to the current routine and an interrupt, four conditions can exist:

   a. Condition 1: No interrupts are present.
      Action: Continue the current routine.

   b. Condition 2: Current routine is of a higher priority than the interrupt.
      Action: Continue the current routine.

   c. Condition 3: Current routine is of same priority as interrupt.
      Action: Continue the current routine.

d. Condition 4: Current routine is of a lower priority than the interrupt.
Action: Suspend the current routine and take whatever measures are necessary for subsequent continuation, then process the interrupt.

4. Interrupt Processing: In processing an interrupt, the system should at least accomplish the following:

a. Since it is impossible for an executing routine to anticipate the instant when an interrupt will occur, the operating system should suspend the execution and save whatever data is required for the suspended routine to continue at a later time.

b. Take necessary steps to delay processing of any interrupt states of a priority equal to or lower than the currently active state.

c. Give control to the interrupt response program for the interrupt currently being recognized.

B. Program Scheduling

1. Levels: The operating system design should be such that the user can assign priority levels for the execution of application programs.

2. Enabling: The application programs shall be able to enable/disable the interrupt system, at least to the extent of controlling interrupt connection

to other application programs.

3. Realtime Clock, Power Fail-Restore: The system interrupt structure should give highest priority to the power failure-restore interrupt, if this option is provided. Next highest priority should be assigned to the system realtime clock. All other peripheral and process interrupts should be of lower priorities.

4. Execution Priority: All programs are assigned an execution priority. When more than one program is scheduled, the comparison of these priorities is the determining factor for one program taking precedence over another. The operating system priority structure should be designed such that interrupts and programs are merged into the overall priority scheme to insure that use of computer time is handled on a priority basis.

5. Execution Scheduling: It should be possible to schedule the execution of a program in one of the following ways:

   a. Interrupt active: when recognized, schedules the execution of a program.

   b. Passage of time: it should be possible to schedule the execution of a program on a timed frequency or at a time of day. The requestor

should be able to specify:

(1) time of day for execution, or

(2) time increment for execution with a starting time, and

(3) execution priority

c. Program request: An executing program should be able to schedule the execution of another program. It should be possible for the requesting program to specify the execution priority of the requested program. With this capability, the requesting program should have the following options available for scheduling the execution of requested programs:

(1) immediate execution (higher priority)-- with optional return to requesting program after completion.

(2) deferred execution (lower priority) -- with no return to requesting program. Execution of requested program follows completion of the requfesting program.

d. Program Suspension: Discontinue program execution for a specified time interval, without disturbing the logical continuity of the program.

C. Memory Utilization

1. Allocation: The operating system should have the capability of dynamically allocating and assigning available core for temporary use to requestors. This allocation should be done according to a requesting priority. When requesting core, the requestor should be able to specify:

   a. amount of memory needed

   b. priority of request for memory

   The operating system should then satisfy requests for memory allocation according to the requesting priority.

   Allocatable memory may be partitioned according to program priorities and high priority memory would not be available to lower priority requestors. The actual dimensions of partitions would be installation parameters and should be done in such a way as to make memory available to satisfy the needs of higher priority requestors first, and thus maintain the overall integrity of the priority scheme.

   It should be understood that a requestor returns allocatable memory, since the operating system cannot sense when the requestor is finished with it.

2. Roll-out: Very often the need develops to execute a high priority program while allocatable memory is

occupied with lower priority programs. When this occurs, the operating system should be capable of suspending and saving the lower priority programs, freeing allocatable memory for higher priority use.

D. Input/Output Handling

1. General: The operating system should include the necessary drivers to operate all I/O equipment (process and non-process) which might be included in the system.

2. I/O Requests: Should have the following characteristics:

a. Consistent format: The requirements for making I/O requests should be organized in a consistent format. The syntactial conventions, which must be followed by requestors should be consistent throughout for all I/O requests.

b. The requestor can specify an I/O device in a symbolic way. It should not be necessary for the requestor to know a device number and other hardware characteristics peculiar to a device, such as channel, line, and equipment numbers.

c. Priority: Requestor specifies the priority at which the request is to be serviced. This is done at execution time.

d. I/O List: Requestor specifies symbolically the location of the I/O list.

e. Completion: Requestor should have the option of

suspending execution of the calling program until the completion of I/O.

f. Buffered/non-buffered: Where available, the requestor should be able to specify that I/O be accomplished either buffered or non-buffered.

3. Processing I/O Requests: The operating system should be the interface between I/O requests and the device drivers. Checking the validity of I/O requests and insuring that they are satisfied in order of priority are functions of the operating system. Conversely, the requestor should not be required to take any action which might involve status checking or I/O timing considerations.

E. Background Processing

1. Definition: A background program is defined in the glossary as a program of no particular urgency with regard to time and which may be pre-empted by a program of greater urgency and priority.

2. Job Processor: If the operating system includes background capability, the job processor should be called or scheduled on a demand (interrupt) basis by a user.

The job processor should have control of all program development, debugging, and other utility programs available to the user.

In addition to having control of vendor-provided utility programs, the job processor should be capable of creating and maintaining a library of background programs generated by the user. The background job processor should, by the nature of the function it performs, operate at the lowest priority.

F. On-Line Diagnostics

1. Definition: Most computer programs employ a measure of dignostics to test for system malfunctions. For this specification, diagnostics considered are those which the operating system performs in order to maintain a properly functioning system.

2. Request: The operating system should verify, in so far as practical, the parameters required by requests to the system. When the system detects an error condition, it should take the necessary action to protect the system and communicate the error condition back to the requestor.

3. Timing: All events should have a timing constraint placed upon them. This would be a parameter defined at load time. The operating system should take action to terminate the event when the time limit is exceeded. This commonly occurs when an I/O device fails and leaves a program suspended, waiting for completion of I/O or when a program

hangs in a loop.

4. I/O: The operating system should provide for suitable alternative action to be taken whenever a primary device fails. Ordinarily, this action would not involve notifying the requestor. However, if the situation is irrecoverable, the requestor should receive some indication of the condition in order that he might take some additional action.
5. Recovery: The operating system should, on regular intervals, make some diagnostic checks, such as checking system constants, to determine that the operating system is intact. It should have automatic recovery capability to be used whenever unusual or abnormal conditions are sensed within the system.

G. Expandability

1. Modularity: The system should be designed and organized with the expectation that it will be expanded. The user should have options available to organize a system to meet his particular requirements while minimizing the inclusion of functions which he does not require.
2. Field Expansion: Field expandable hardware components should be supported by comparable operating system expandability.

V. SYSTEM GENERATION

A. Introduction

System Generation (definition):

"The task of establishing an operational system composed of:

a. installed hardware,

b. vendor-supplied programs, and

c. those programs written expecially for the application, which will be expected to function in either foreground or backgroud mode on the application computer, while the process is on-line."

Certain utility, diagnostic, or various other categories of programs and any special hardware required by them may also be operational in the background mode or when the application computer is off-line.

Program Development is the series of steps between defining the problem and having a functioning program. Testing and debugging begin when the program has been prepared in source form.

Some programming errors are detected when the program is compiled; others require execution of the program, possibly with simulated inputs and outputs, traces and dumps, before yielding to detection.

When the program functions as intended, it is then installed into the system. There will usually be changing system requirements which preclude permanent installation of a program. This sequence may be repeated several times, as part of the debugging process.

This section will not deal with the way the compiler for the high level language(s) transforms the source statements into executable object code. However, the treatment of all source statement types by the compiler as related to the administration of the object code, such as converting (data), loading, linking and relocating, are dealt with as deemed necessary.

B. Software Development Considerations:

1. User Application Programs

a. Program Generation

(1) Coding or statement preparation;

(2) Source input preparation (i.e., key punching; etc.);

(3) Compile/assemble

b. Data Preparation

(1) For On-Line use

(2) For testing use

c. Testing: Both partial and complete systems

(1) Loading (and linking as required);

(2) Simulation of process;

(3) Tracing:

(a) Arithmetic

(b) Logical

(4) Debugging

2. Vendor-Supplied Programs

Customizing vendor software to needs of a specific application. The program-development programs: compilers, assemblers, loaders, emulators, etc. The on-line programs, operating systems, scan packages, etc.

a. Remove modules which are not applicable, device drivers not used, conversions not used, ect.

b. Supply special information about hardware configuration (see also D, Linking Software to Hardware). Examples are:

core size;
disk capacity;
tape configuration;
input options;
output options;
interrupt trap addresses;
special indexing addresses;
other special use locations;
data acquisition options (direct memory access, etc.);
signal conditioners
(1) logical inverters
(2) shifters

C. Loading

1. There should be the ability to generate object coding with appropriate "header" information so that subsequently the object code can be:
   a. Loaded in core as compiled with minimal instruction from its header;
   b. Loaded in next available core (considering pages, odd, even, etc.);
   c. Loaded in core after a halt pending an on-line origin directive;
   d. Loaded in object-image onto secondary memory as directed in header;
   e. Loaded as in 'd' except in "next available segment" of secondary memory;
   f. Loaded as in 'd' except after a halt pending an on-line origin directive.
2. There should be the ability to copy:
   a. Relocatable program images from secondary memory into reloadable form on compatible input media;
   b. Data images from secondary memory into reloadable form on compatible input media.
3. There should be provisions to reload previously dumped relocatable program images and data images into secondary memory beginning at specified starting locations.

4. There should be provisions to load compiled relocatable program images and compiled or assembled data images onto secondary memory at a suitable starting location.

D. Linking Software to Hardware

Ability to inform the system through a mechanism, such as a symbol table available to the compiler, of:

1. The physical termination identification of each peripheral industrial application component (process I/O device) with which it must communicate: (sample specification statement)

   Program Identifier = Physical Termination Identifier (ID)

2. The physical identity of the communication channel(s) over which information will be exchanged with each industrial applications component: (Sample specification statements)

   Program Identifier = (industrial application component) via (communication channel, identifier)

   Data to/from (industrial application component) via (communication channel, identifier)

3. The physical identification of:

   a. each separable interrupt level and its program identifier;

   b. each separable interrupt point at a given level and its program identifier.

E. Language-to-Language Linking

In order to provide maximum flexibility to the programmer, he should be allowed to choose the language best suited to stating the task to be performed. He should be permitted to interix these language statments within one program.

1. Industrial Process Language to Assembler;
2. Industrial Process Language to Problem Oriented Languages;
3. The Industrial Process Language Complier should tolerate "Foreign" statements in the above languages and permit the Assembler and other Compiler to analyze these for correctness, etc.

F. Areas of Improvement

1. Greater modularity of software

   a. Operating System Software

   b. Other Packaged Software

   The vendor documentation of these packages should be adequate for determining how to remove task modules and associated subroutines which the user may not require in his system.

2. There should be a two-way-entry method of determining whether a subroutine can be dropped from the library, i.e., by subroutine, what tasks use it, and by task, what subroutines are used.

Examples of modules are:

a. I/O drivers

b. Data conversion routines

c. Scan programs

d. Format conversions

e. Arithmetic subroutines

f. Parameter-passing options

3. A system generation scheme is required for small object systems, such that the object system can be generated on some other, larger system. The larger system may be located locally, or remote from the object machine, and may be of a different make or model from the object machine.

VI. SYSTEM MAINTENANCE

A. General

Reliability and maintainability are interacting characteristics in a system and result in a single feature: system availability. The implementation of system maintenance depends on the availability required of the system, and the extent of the task depends on the reliability of the component parts. In the sections that follow, the implementation of maintenance is ommitted on the basis that it is peculiar to a system and must be developed for that particualar system.

The objective will be to organize a preventive maintenance program where all maintenance is on a scheduled basis. Nevertheless, provision should be made for diagnostic aids to allow rapid replacement of failed equipment. The remainder of this section assumes the use of planned preventive maintenance.

B. Hardware Maintenance

1. C.P.U. and Standard Peripherals

Diagnostic testing programs shall be available for the C/P.U. and standard peripherals. As this is the basic vendor-supplied equipment, this can be regarded as a mandatory requirement. It is desirable that two forms of test be available:

a. Simple on-line fault indicating programs.

b. Comprehensive off-line fault finding programs.

2. Process Equipment

Diagnostic programs should be provided for the process equipment under computer control. Basically, this may be an equipment logging and monitoring system. As a minimum, the faulty control loop should be indicated, and in some circumstances it may be desirable to pinpoint the faulty equipment, i.e., valve, sensor, etc.

3. Economic Preventive Maintenance

To ensure that the preventive maintenance program is carried out economically, it is suggested that the logging and monitoring system be used to develop operational reliability data on the installation and that this data should be used to modify the maintenance shcedule.

4. Computer Assisted Maintenance

It may be desirable to implement, on a computer, programs that will supply or process data to assist maintenance personnel in their tasks, e.g., list likely equipment faults, suggest the type of maintenance--repair or replace, etc.

5. Spare Parts Inventory

This could usefully be performed by a computer. Stock levels to be held could be produced by the computer, based on quantity in use, past reliability history, and order lead time.

C. Software Maintenance

Adequate provisions must be made to protect software integrity. Controls should be built into the system to restrict the ability to destroy data or programs. Similarly, the incorporation of program revisions or updates must be organized to reduce the possibility of erroneous or incompatible software existing in the system.

The implementation of software to handle the man-machine interface should be such as to minimize the possibility of the operator carrying out unauthorized operations (while this is primarily needed for product and personnel safety reasons, improper operation of equipment may reduce reliability). If it is possible in a system to reroute a device, (e.g., a computer output fails and there is a spare output available), then it should be possible to easily amend the reference to that device.

On-line routines shall be available to allow initialization of replacement equipment.

D. Documentation

Basic to any maintenance function is the system documentation. Note also that this documentation must be maintained to reflect the current status of the hardware and software throughout the the life of the system.

The following list constitutes a minimum documentation package that should be provided with a system:

1. System Description and Objectives
2. Reference Manuals
3. Design Specifications
4. Acceptance-Testing Documentation
5. Warranties
6. Hardware Maintenance Handbooks
7. Contract Maintenance Schedules (if used)
8. Spare Parts Lists, Prices and Sources
9. Programming Manual
10. Program Descriptions
11. Program Listing and Flow Charts
12. Initializing Procedures
13. Operating Procedures
14. Program Sources (Users' groups, authors, etc.)

VII. CONTROL COMPUTATION

Means should be provided for mathematical computations including at least the operators for addition, subtraction, multiplication, division and exponentiation and for logical operations and decisions. The remainder of the section is primarily aimed at specifying functions necessary for a problem oriented language. The underlying requirement for this section is that a knowledge of the computer operating system should not be necessary to construct control schemes.

A. Control Loop Servicing

1. It should provide control at a frequency corresponding to the frequency of acquisition of input data for a control loop.
2. It should provide control at a frequency corresponding to a sub-multiple of the frequency of acquisition of input data for a control loop.
3. It should provide control at a frequency independent of the acquisition of input data.
4. Control computation for a computer control loop should be such that the time between the scan and the control output for a given loop is at a minimum.

B. Control Loop Generation

Control loops should be constructed using modular function blocks.

a. libraries of control algorithms, limit check

algorithms, filter algorithms and output algorithms should be provided.

b. both incremental and positional control algorithms should be available.

c. the user should have the ability to construct algorithms beyond these supplied with the system.

C. Data Base

A structured file should be provided for specification or storage of control loop information such as:

a. input variables

b. messages

c. common data

d. constants

e. control loop cascading specifications such that:

1. the operator may connect or disconnect cascade at any level

2. it should be possible to connect or disconnect cascade on request from a higher level

3. should the intermediate level of cascade be disconnected, higher levels may optionally be disconnected as well

A means of easily accessing and modifying information in the file should be provided through the operator's I/O devices, the system I/O devices and high level programs.

VIII. INTERCOMPUTER COMMUNICATIONS

A. General

1. Hierarchical systems, vertically organized, will consist of "mother" and "daughter" systems. The "daughter" systems will be defined as those of higher dynamic requirements, and will usually be sensor-based, and smaller in size and cost than "mother" systems. "Mother" systems will ordinarily be management-oriented, but may also be involved in direct operation of production or testing equipment.
2. Parallel systems, horizontally organized, will consist of either "mothers" or "daughters", with approximately equal dynamic requirements.

B. System Discipline

1. System integrity is of primary importance so means must be provided for error detection. Failure of one element, such as a daughter component, should not cause a cascading of failures throughout the hierarchy. Local power failures or other failures must be noted by adjoining elements, even though normal data communications (or control of communications) may be one-way.
2. System generation methods will be used to structure the hierarchy, to determine the source

of control of data transmission for other than fault conditions.

3. Logic must be provided to synchronize the time of day between computers.

4. Illegal requests, such as attempts to write into protected areas, or to overflow buffer boundaries should be detected and prevented.

C. Hardware Capability

1. A minimum capability shall consist of ability to transmit or receive, half-duplex.

2. Capability shall be provided for at least binary and ISO, or binary and ASCII.

D. Software Capability

1. The same general procedures are to be used, whether the computers are local or remote from each other. System generation should accommodate timing restrictions due to distance and transmission modes for each channel used.

2. Software for data transmission should not distinguish between transmission of data and transmission of programs.

   a) Capability should be provided for transmission of either random or sequential data.

3. Data transmission to other computers should be handled in a manner similar to methods used for handling other peripherals in the system.

4. The ability to output messages to a remote computers's peripherals is required. This type of request should take the form of standard I/O statements.

5. The option to either continue a program calling for data transfer or waiting until the transfer is completed should be provided.

E. Remote Task Selection

1. A provision must be made, for remote control of computer task selection. For example, data received may be intended for a file, or it may be a program or request for a program to be executed. System monitors must allow a means to examine a completed data transmission to determine the end-use of the data.

2. Transmission Control Characters

a) A standard control character or control character string format must be defined as part of the implementation of high level industrial computer languages.

IX. FILE MANAGEMENT

A comprehensive system for management of the large amounts of data (including program files) should have the following characteristics:

1. The ability to define files to the operating system while causing their creation by means of a maintenance or utility program.

   The following items should be available as defining parameters for creation of files:

   a. Label: either a number or character string.

   b. Device on which the file will exist

   c. Type of file:

      (1) Organization:

          Sequential access, or

          Random acess

      (2) Record type:

          Fixed length, or

          Variable length

      (3) File type:

          Fixed length, or

          Dynamically expandable

   d. File Size

      (1) number of records for a fixed length, or

      (2) initial number of records, plus incremental number of records, plus maximum allowable number of expansions (increments) for dynamically

expandable files.

e. Record Size

(1) number of words/characters per record, for fixed length records, or

(2) maximum allowable number of words/characters, for variable length records,

(3) Blocking factor

f. File integrity

(1) Dynamically definable protection via "open" and "close" types of executable statements of application programs.

(a) "Open" file, with the following items as parameters:

File name or number

Definition of status as:

Reserved for exclusive use of "opening" program. (puts file into "protected" status for all other programs)

"WRITE" reserved for "opening" program (places file into "read-only" status for all other programs)

Unlimited access for any program

Error return, for the following conditions:

file nonexistance

file currently open with a conflicting status.

(b) "Close" file, with the following items as parameters:

File name or number

Error return conditions as follows:

File nonexistant

(2) Fixed definition of protection for life of file, with the following options:

(a) Read-only by any program except utility/maintenance system

(b) Unprotected - available for read/write use by any program.

2. The ability to remove or <u>delete</u> files by means of a maintenance or utility program, in such a manner that inadvertant deletion is difficult.
3. The ability to <u>copy</u> files from the device on which it resides to another device without destroying the original file, by means of a maintenance or utility program.
4. The ability to <u>optimize</u> location of storage allocated to files by means of a utility or maintenance program e.g. moving files, not stored contiguously, into continguity.
5. The ability to <u>access</u> logical records in a file by means of the following types of executable statements

in application programs, with return of error information to the application program (in such cases as: file non-existance, attempted acess of a closed file, attempted write to a "read-only" file, access attempted beyond range of file, and attempted access of a "protected" file);

a. "READ" a logical record from the file into a variable or list of variables specified in the statement. (Subject to file integrity specifications)

b. "WRITE" a logical recrod from a variable or list of variables specified in the statements, to a file. (Subject to file integrity specifications)

Definitions:

FILE: An organzied structure, consisting of an arbitrary number of records, for storing information on a bulk storage device, e.g., disk, drum, core, tape.

RECORD: A segment of a file consisting of an arbitrary number of words/characters.

Logical Record - A record organized according to its application, treated as a logical entity.

Physical Record - A record sized for efficient transfer of information between devices, and/or efficient

utilization of storage space on a device.

BLOCKING FACTOR: the number of logical records per physical record.

SECTION II

FUNCTIONAL REQUIREMENTS FOR LANGUAGE FEATURES FOR A PROCEDURAL LANGUAGE FOR INDUSTRIAL COMPUTERS

Subsequent to the appearance of the general functional requirements reproduced just previously, the Long Term Procedural Languages Committee modified and extended them specifically for the LTPL. These are the "green sheets" which are frequently discussed in later Minutes of the Workshops. They were published on pp. 77-101 of the Sixth Minutes of the Purdue Workshop on Standardization of Industrial Computer Languages.

PRECEDING PAGE BLANK-NOT FILMED

FUNCTIONAL REQUIREMENTS FOR LANGUAGE FEATURES FOR A PROCEDURAL LANGUAGE FOR INDUSTRIAL COMPUTERS

(Adopted by the Workshop on Standardization of Industrial Computer Languages - October 29, 1971)

INTRODUCTION

As functional requirements, these facilities must be considered to represent a minimum capability which must be available to the user of the language; other facilities will surely be provided as language design ensues. Additional functional requirements will result from work on acceptance criteria, hardware/software linkage and program management. (It should be noted that the practical impact of functional requirements is that they represent design criteria to which the language designer must address himself).

During the development of the functional requirements for tasking, interrupts, events, and semaphores, it was necessary to develop fairly formal execution models to understand the functional requirements; these models are included in the explanation of these functional requirements and are not as a required form of implementation. For example, explanation of tasking for the user might be possible with a simpler model.

PRECEDING PAGE BLANK-NOT FILMED

1.

Interrupt Handling And the Synchronization Of Parallel Computational Processes

INTRODUCTION

Effective handling of interrupts and asynchronous and parallel computational processes is a primary requirement for a real-time language. This section outlines those facilities required for a procedural language for real-time industrial computers for interrupt handling and synchronization; another section will outline tasking requirements.

Execution Model

In order to determine functional requirements, we will first explain the general execution model assumed. (The attempt is to make this discussion "free standing" - definitions are from Multics).

A sequence of statements which may be executed by the CPU is a segment. A computational process exists when a processor is executing instructions from some segment. Many computer systems have the capability of executing several computational processes simultaneously. This may happen on a single processor system when one process must wait for an I/O operation, for example, and another process can proceed with useful work. When this happens, parallel computational processes are occurring.

In general, parallel processes proceed with no defined relationship between them and thus are asynchronous. Often, however, the occurrence of some programmed action within a computational process is of interest to some other computational process. The occurrence of this programmed action then is considered an event. Since its time of occurrence is, in general, undefined outside the process which caused it, in general it is also asynchronous. At times it is required to maintain some predefined sequential relationship between programmed actions in otherwise asynchronous parallel computational processes. This is called the synchronization of parallel processes.

In a computer system it is also possible to have an occurrence which is not a programmed action within any computational process, and which requires some response from the system. Such an occurrence is called an interrupt.

The general organization of the systems upon which this language is to be used is assumed to include software responsible for operating the computer system, often called the executive or operating system, and the application software which will be run as (possibly parallel) computational processes.

Direct interrupt handling is considered to be within the province of the operating system, and any language proposed for interrupts will not be used in applications programs. The event language will be used to provide real-time operation of applications programs through an event supervisor in the executive.

FUNCTIONAL REQUIREMENTS

A user of a procedural language for the real-time programming of industrial computers must have the following capabilities:

Interrupts

- Ability to indicate that some segment is to be executed upon the occurrence of some particular interrupt.

Events

- Ability to retain the flow of control at a point in a computational process until some event has occurred.
- Ability to cause the occurrence of an event at some point in a computational process.

Synchronization

- Ability to synchronize parallel computational processes with security.

2.

TASKING FACILITIES

INTRODUCTION

The purpose of this section is to propose a set of functional requirements for the tasking facilities of the LTPL. First a number of terms are defined (in the Appendix). These terms are used to describe a very general model of tasking. The next section proposes a set of functional requirements for the LTPL in terms of this general model. This proposal combines the valuable features of all the previous proposals on the subject into one unified facility. The last section very briefly discusses the relation of these proposals to existing or proposed languages.

MODEL

This section is an attempt to define a model for the operation of a task and the interaction between tasks. This model is intended to be rich enough to include all the functions available in many existing and proposed systems, although most systems do not contain all the functions available in the model. Later in this paper an attempt will be made to reduce the model to a more compact form that satisfies the functional requirements for a procedural language for industrial computer applications.

A task can be in one of a number of states. In some states additional information is required, such as the segment which the task is executing. The state of the task and the additional information required for particular states is contained in a portion of storage called the state vector of the task. This state vector is known to the system and is used by the system to control the execution of the task.

Figure 1 illustrates the possible states and some of the transitions that could take place between them. In principle, a transition is possible from any state to any other state, but some of these transitions are of little practical value. The meanings of the various states are as follows:

Ø. Nonexistent

The task is not known to the system in any way; i.e., no storage has been allocated to contain the state vector for the task. (Since the task does not exist, this is not really a "state", but considering it as such clarifies the discussion).

BEST AVAILABLE COPY

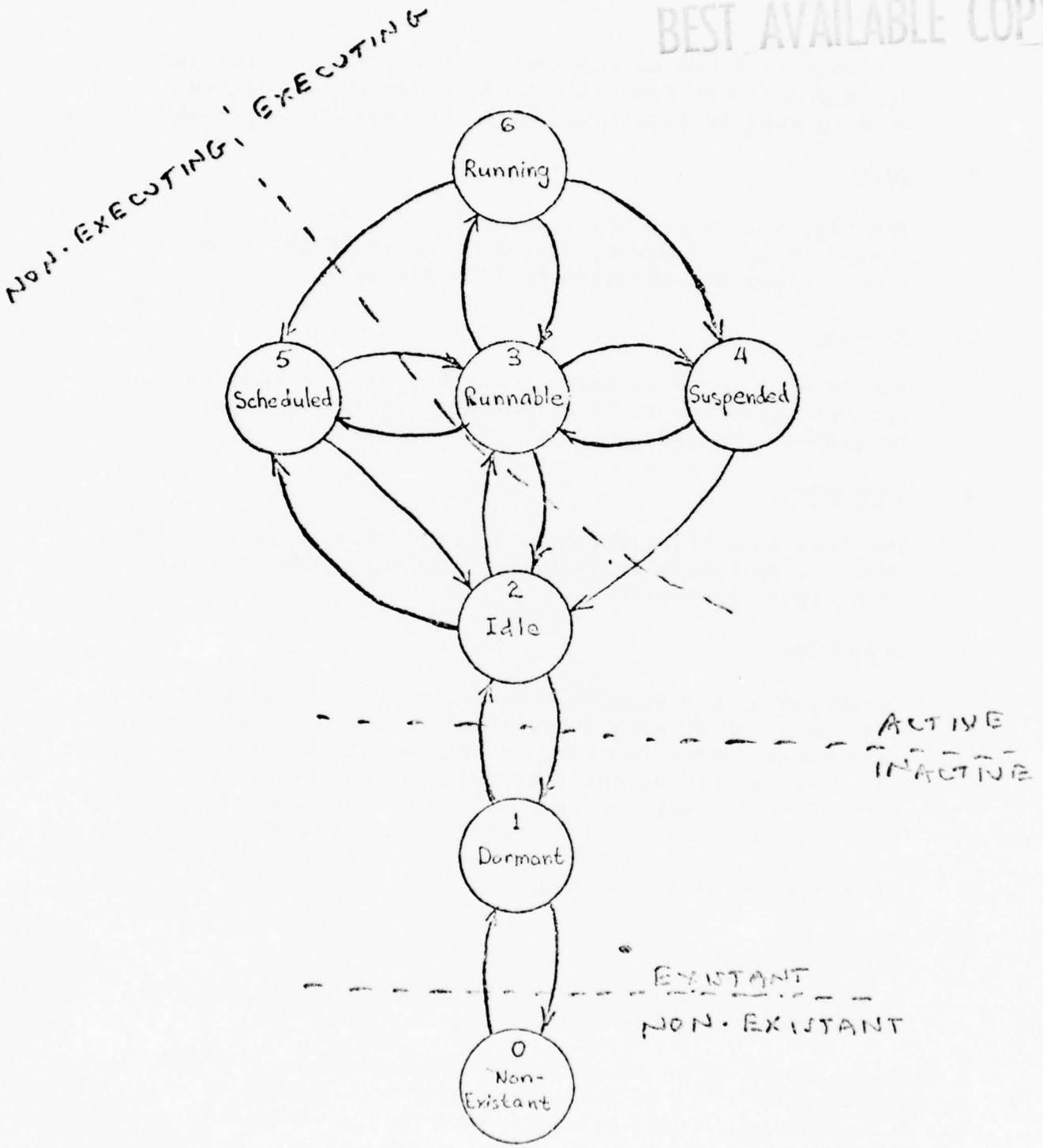


FIGURE 1.

Task State Diagram
With Some
Possible Transitions

1. Dormant

   The task is known to the system; i.e., storage for its state vector has been allocated. However, no segment or data area is associated with the task in this state.

2. Idle

   The task exists and has a segment and a data area associated with it. However, execution of the task is neither taking place or scheduled to take place.

3. Runable

   The task is ready to execute and has all requested resources allocated to it except for a physical processor to execute its code.

4. Suspended

   The task is waiting for some event to occur, such as for a resource to become available or for some other task to give it a signal to continue.

5. Scheduled

   Execution of the segment associated with the task is scheduled to occur at some future time or on the occurrence of some event. When the time is reached or the event occurs, the task is automatically transferred to the runable state. The main difference between scheduled and suspended is that the scheduled state exists before execution of the segment has started, whereas the suspended state occurs in the midst of execution of the segment.

6. Running

   A physical processor is actually executing the segment associated with the task.

A task exists if it is in any state except state Ø. A task is said to be active if it is in any of the states 2 through 6, i.e., if it has a segment and a data area assigned to it. A task is executing if it is in states 3, 4 or 6. Thus, a process can be described as all the actions that a task performs during the time period between a particular entrance to the executing state and the next exit from the executing state.

The transition from one state to another can be caused by an action of the task itself or by an action of another task or by some action of the system. In order to clarify the meaning of the state diagram, it will be helpful to explain the actions required to cause some of the more important transitions.

A task comes into existence when a state vector for it is allocated and this state vector is made known to the system; it ceases to exist when the state vector is de-allocated or when the system ceases to be aware of the state vector. A task becomes active when a segment and a data area are assigned to it, and inactive when the segment and data area are released. A task moves from the idle state to the scheduled state when the system is informed that the task should be executed at a particular time or on the occurrence of some event. The task starts executing when a scheduling condition is met or when another task requests its execution.

The reason for having both the idle and the dormant states is to allow for a distinction between the creation of a tasking and assigning a code segment to it. In some languages both of these functions are performed by the declaration of a task, i.e. the declaration puts the task into the idle state. In other languages the segment may not be specified until the task is put into either the runable or the scheduled state, and therefore the declaration puts the task in the dormant state.

A task starts executing by entering the runable state. Transitions between states 3 and 6 are caused automatically by the system and are not under the direct control of any task. At any instant in time, any number of tasks can be runable, but the number of running tasks is limited to no more than the number of physical processors available to the system. The mechanism by which the system decides which task should be run by a given physical processor at a particular time involves a numerical value which is part of the state vector and is called the priority of the task. When a physical processor becomes available, the system allocates it to the highest priority runable task. If several runable tasks have the same priority, the system makes an arbitrary choice. Some systems may only have one priority level, but this is generally not useful in an industrial computer application.

If a task wishes to change its own state vector, it does so by making a request to the system. While this request is being processed, the task is still considered to be in the running state.

Therefore, the only transition which a task can cause on itself are the transitions from state 6 to one of the other states. All other transitions can be caused only by the actions of other tasks or of the system. In principle, a task can request a transition of itself from state 6 to any other state, and can request any possible transition for another task. In a practical system, however, only some of these possibilities will be allowed.

One of the more important states in the model is the suspended state, state 4. Where a task is suspended, it is not performing any actions but is waiting for something to happen outside itself. When the awaited thing happens, the task becomes runable again. While a task is suspended, it can be waiting for one of the following things:

1. A time interval to elapse.
2. Another task which requested its suspension to allow it to continue.
3. An event to occur.
4. A semaphore which it is trying to secure to be released by another task.
5. A resource controlled by the system to become available.

Some systems may allow a task to be waiting for more than one thing at the same time. If this is allowed, there must be a set of rules to determine what combination is required to make the task runable.

Although this paper is not intended to deal with interrupts, a brief discussion of the relationship of interrupts to this model is appropriate. There is no mechanism for an interrupt to affect the state of a task directly. However, recall that a task can enter the runable state from either the scheduled state or the suspended state on the occurrence of an event. It is possible for a system to provide a facility for establishing a connection between an interrupt and an event. Once this connection is established, the system will automatically cause the event to occur when the interrupt is received. A facility for establishing and controlling such connections should be provided by an industrial computer system.

To summarize, then, a task is defined by a state vector which consists of the following components (some of which may be empty at various times):

1. A state number.
2. The name of a segment.
3. The name of a data area.
4. A priority number.
5. A list of scheduling conditions.
6. A list of things for which it is waiting.

Three things can change the value of the state vector: the task itself, another task, or the system.

FUNCTIONAL REQUIREMENTS

Once the model of a task has been defined, it is possible to proceed to consideration of a language that will provide those parts of the general model which are required for industrial computer applications. This must be done in two stages: first, select those parts of the model that are required by the application, i.e., define the functional requirements; second, define language features that will satisfy these requirements. This paper will only attempt the first of these two steps. This section will propose a subset of the general tasking model as the functional requirements for a procedural language. These functional requirements must be considered by the LTPL Committee, and only when agreement on them is reached, we can proceed to defining the language features.

The specific functional requirements proposed here have been distilled from all the previous work done by the participants in LTPLC from throughout the world.

In terms of the tasking model presented above, all that is needed to specify the functional requirements for a particular application is to specify which state transitions are allowed and what information must be supplied when one of the transitions is requested. The transitions which can be caused by a task for itself, for another task, and by the system must be separately specified.

The following paragraphs define the functions required in each of the three areas. Each function is given a name written in lower case and underlined. These names serve only as labels for functional capabilities are are not intended to have any relation to possible linguistic representations.

A task is made known to the system with a declaration, which assigns a name to the task and causes a state vector for it to be statically allocated. This does not actually cause the task to exist; the task is brought into existence when the system performs the create function. This occurs when the block containing the declara-

tion is activated; i.e., when the static declaration is invoked. The create function puts the task in the dormant state. The system can return the task to the nonexistent state when it performs the delete function. This occurs when the static declaration ceases to exist; i.e., when the block containing the declaration is deactivated. This task can be made available to other blocks by making its name available through the normal language rules for scope of names or by passing it as an argument to a procedure.

Once a task exists it can be activated by another task with either the execute or the schedule function. In either case, a segment name must be supplied. The data area is supplied automatically by the system through activation of the outermost block of the segment. In the case of the execute request, the task enters the runable state. The schedule request puts the task in the scheduled state and requires that scheduling conditions be supplied. These conditions can consist of an initial time, a repetition interval, and/or an event. When the scheduling conditions are met, the system makes the task runable with the ready action. Either the execute or the schedule request may optionally specify an event which is to occur when the task stops executing.

One task can suspend another with either a delay or a suspend request. The delay function must specify a time period. The object task will be suspended for that period, after which it automatically becomes runable again. The suspend request will leave the object task suspended until some task makes it runable again with a continue request.

A task can stop the execution of another task with the terminate function. If the object task still has scheduling conditions in its state vector, it will enter the scheduled state; otherwise it will become dormant. The exit function will produce the same results except it applies to the task that executes it itself. The system can also stop the execution of a task with the same results by performing the abort action.

A task can make another task dormant regardless of any scheduling conditions by using the kill function. This clears out any scheduling conditions which may have existed. The system can do this with the exterminate action. A task can force itself to become dormant with the killme request.

Once a task is runable, the system can put it in the running state with the run action. The pre-empt action by the system returns the task to the runable state.

A task can suspend itself with either the delayme or the wait function, which must specify either a time interval or an event, respectively. A task can also be suspended if it executes the secure function on a semaphore that has already been secured. In this case, it will become runable again when another task executes a release on the same semaphore.

If a task requires the use of a resource which is controlled automatically by the system and that resource is unavailable, the system will suspend the task with the inhibit action. The system can make a suspended task runable with the unsuspend function, which will happen if the task was waiting for a time interval, an event, or a system resource and the awaited condition becomes true. If the task is waiting for an event, another task can make it runable with the signal function specifying that event.

There are also several functions that do not directly change the state of a task, but do affect the other information in the state vector. A schedulome request will put a scheduling condition into the state vector. The priority of another task can be set with priority, and of the requesting task with mypriority. Similarly, the entire state vector of a task can be examined with either status (for another task) or mystatus (for the same task).

Table 1 summarizes all the functional capabilities required for tasking. This table numbers each function, shows the effect on the state vector, and specifies the information that must be supplied for each. For all the functions that require a task as an argument, an error occurs if that task is not in the state specified on the left-hand side in the state change column. Figure 2 shows all the functions on the state diagram, with each transition labeled with the appropriate function numbers from Table 1.

Although the list of functional capabilities looks very complicated at first, it is much simpler than the complete model. Note that state 2 is not used at all, and only about half the possible transitions are used. Also, analogous operations are defined as separate functions depending on whether they operate on the current task or another task. However, these pairs of functions could each be implemented with a single language feature. For instance, a single DELAY statement could be used without a task name for delayme and with a task name for delay. Similarly, if the system were implemented in the same language, some of the system functions could be implemented with the same language features as some of the non-system functions. Furthermore, although a lot of functions have been defined, each one represents a fairly simple operation on the state vector; therefore they should not be very difficult to implement.

Since the idle state in Figure 2 has no transitions entering it or leaving it, it appears to be unnecessary. However, it is useful to retain this state in the model so that the model is still powerful enough to show the features of all the proposed languages and so that it will be still possible to consider proposals to add functional requirements that require the use of the idle state.

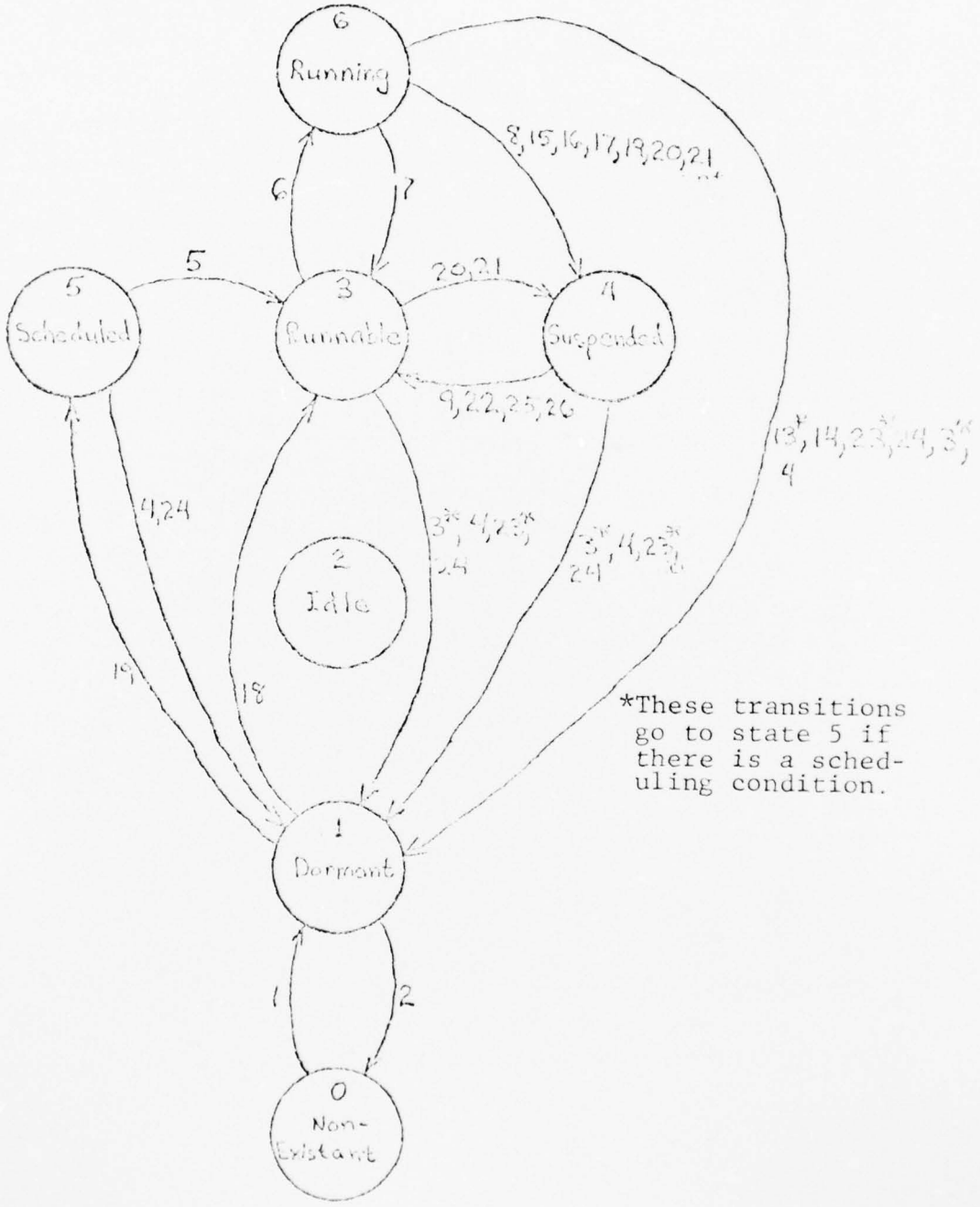
BEST AVAILABLE COPY

	FUNCTIONAL OPERATION	STATE CHANGE OF OBJECT TASK	ARGUMENT REQUIRED
System Actions	1. create	∅→1	<task>
	2. delete	1→∅	<task>
	3. abort	3,4,or6→1 or 5	<task>
	4. exterminate	3,4,5,or6→1	<task>
	5. ready	5→3	<task>
	6. run	3→6	<task>
	7. pre-empt	6→3	<task>
	8. inhibit	6→4	<task>
	9. unsuspend	4→3	<task>
Operations by a Task on Itself	10. scheduleone	adds to schedule list	<time>|<event>
	11. mypriority	sets priority	<priority level>
	12. mystatus	returns state vector	
	13. exit	6→1 or 5	
	14. killme	6→1	
	15. delayme	6→4	<time>
	16. wait	6→4	<event>
	17. secure	6→4 or 6	<semaphore>
Operations on Another Task	18. execute	1→3	<task>,, <data area>
	19. schedule	1→5	<task>, {<time>|<event>}
	20. suspend	3 or 6→4	<task>, , <data area>
	21. delay	3 or 6→4	<task>, <time>
	22. continue	4→3	<task>
	23. terminate	3,4,or6→1 or 5	<task>
	24. kill	3,4,or6→1	<task>
	25. release	4→3	<semaphore>
	26. signal	4→3	<event>
	27. priority	sets priority	<task>, <priority level>
	28. status	returns state vector	<task>

TABLE 1.

Summary of Tasking Functions

BEST AVAILABLE COPY



*These transitions go to state 5 if there is a scheduling condition.

FIGURE 2

STATE DIAGRAM OF
PROPOSED TASKING SYSTEM

3 DATA TYPES AND MANIPULATION - FUNCTIONAL REQUIREMENTS

1. All data can be initialized.

2. The only implicit data type conversion occurs during assignment or with I/Ø operations.

3. Problem data must include arithmetic and string data.

4. Program control data must include data types required to support tasking, events, interrupts, I/Ø, synchronization and flow of control.

5. Hardware/Software data types for interfacing hardware and software should be provided as required.

6. Arithmetic data will include integer and float data types.

7. String data must include character strings and bit strings. There will be both fixed and varying strings. Maximum string length is implementation dependent.

8. Data are organized as
   a. Scalar Data (single items)

   b. Data Aggregates

      . Arrays (wherein all elements of the array are of the same type)

      . Structures (all elements do not have to be the same type).

9. There shall be a mechanism for accessing any particular element of a data aggregate.

10. There shall be facilities for manipulation of problem, program control, and hardware/software linkage data. Operators will include

    + - prefix
    + - infix
    * /
    **
    and, or not
    comparisons
    concatenation

4.

STATIC PROGRAM STRUCTURE

INTRODUCTION

Necessary to any procedural language is a means of specifying the structure of the program. This paper addresses static program. These are the facilities needed to define a collection of statements to be a program, specify data and data structures and manipulation of that data.

DEFINITION

In discussing data types, problem data means data items such as fixed-point variables. Program-control data is utilized by the tasking and interrupt handling facilities.

FUNCTIONAL REQUIREMENTS

A programming language for real time industrial computers must provide the following capabilities.

DEFINING A MODULE

- Means to identify a group of statements to be treated as a module. (Such as an in-line function or a subroutine).
- Means to identify a module or group of modules as a segment. A segment would be available for tasking and interrupt handling purposes.

SPECIFYING DATA TYPES

- The ability to declare names and attributes of problem data, program-control data and hardware/software linkage data.
- Explicit declaration shall be required where standard defaults or contextural declaration cannot supply reasonable, understandable attributes.

STATEMENTS

statements which provide:

- Ability to assign values to data items.
- Ability to specify operations to be performed with data items.
- Ability to call a subroutine.
- Ability to invoke an in-line function.
- Ability to specify several statements as a group for looping or iterative operations.
- Conditional and unconditional branching for altering the flow of control within a program.

MODULARITY

- The language shall facilitate the linkage of separately compiled modules.
- The language must provide the ability to specify the scope of data-items within and outside of the modules.

5) I/O REQUIREMENTS

In a programming language for realtime applications it is necessary to have available:

1. A sufficiently powerful set of I/O statements and formatting possibilities for input and output of numeric and alpha-numeric data which is not too complex and complicated to be efficiently implemented.

2. A possibility to transfer unformatted data e. g. to and from backing store or between computers in a computer network.

3. Tools for particularily process-oriented data transfer with special emphasis on:

   3. 1. efficient use of process control hardware

   3. 2. the possibility for the programmer to add new types of I/O - facilities for new and non-standard I/O devices

   3. 3. Inter-system transfers which do not touch the CPU or even the main computer

   3. 4. the ability to efficiently program I/O which involves a data transformation related to the I/O operation (e.g. conversion from analog counts to engineering units)

4. I/O shall be defined in a way that facilitates the construction of new higher level I/O - functions from the basic ones (e.g. appropriate synchronization mechanisms should allow the execution of an output and the respective input response in one indivisible operation).

5. The language with facilitate graphics I/O.

6. The language should facilitate efficient file handling.

7. The user shall be able to define error recovery for I/O operations.

8. The definition of I/O must not lead to an implementation which conflicts with the tasking structure of the user program.

6.

DYNAMIC PROGRAM STRUCTURE

Functional Requirements

1. The language shall support multiple and nested re-entrancy upon user declaration.

2. The language need not support recursion.

3. The language shall automatically support allocation of storage for modules declared re-entrant.

4. There shall be explicit means to dynamically allocate storage.

5. It shall be possible to activate a task whose continued activation is not dependent upon the continued existence of its activator.

6. The language shall have facilities for transmitting data from one task to another.

7. The language shall have facilities for sharing data between tasks.

7.

PROGRAM MANAGEMENT FUNCTIONAL REQUIREMENTS

1. Features for software visibility

Contents of programs should be visible through various tools such as listings and messages during program preparation, debugging, and maintenance. Features of documentation useful for this purpose are as follows:

(1) Source program listing

(2) Intermediate language listing

(3) External references

a) Routine entry names

b) Data names referred from external programs

c) External routine names

d) External data names

e) Common data names

(4) Symbol table

(5) Physical memory map

(6) Program unit name listing

(7) Program structure listing (phases, local subroutines, etc.)

2. Compile time facilities

It is desirable for the language to have some compile time facilities which contribute to the flexibility of language through conditional compilation. These facilities should be, however, investigated carefully because they will introduce a pre-compilation phase to the compiler and it may be too much for small size computers.

3. The language design shall give careful attention to the provision of debugging features such as:

   Setting of conditions for debugging operations
   Executable statements for debugging operations
   Auxiliary listings useful for debugging
   Error messages
   etc.

8.

Hardware - Software - Linkage

Functional Requirements

1. Ability to inform the system through a mechanism, such as a symbol table available to the compiler, of:

   a) The physical termination identification of each peripheral industrial application component (process I/O device) with which it must communicate: (sample specification statement)
   Program Identifier = Physical Termination Identifier (ID)

   b) The physical identity of the communication channel(s) over which information will be exchanged with each industrial applications component: (sample specification statements)
   Program Identifier = (industrial application component) via (communication channel, identifier) Data to/from (industrial application component) via (communication channel, identifier)

   c) The physical identification of:

      1. each separable interrupt level and its program identifier

      2. each separable interrupt point at a given level and its program identifier

2. The requestor can specify an I/O device in a symbolic way. It should not be necessary for the requestor to know a device number and other hardware characteristics peculiar to a device, such as channel, line and equipment numbers.

3. The language must facilitate the ability to switch devices in a secure way in case of failure.

GENERAL DEFINITIONS

Processor -	A machine capable of executing program statements.
Segment -	A sequence of statements executable by a processor.
Data Area -	A portion of storage used during the execution of a segment (The storage is not necessarily contiguous or of fixed size.)
Computational Process -	An instance of execution of a segment by a processor using a data area.
Parallel Computational Processess -	Computational processes which occur simultaneously.
Synchronization of Parallel Computational Processes -	Controlling the execution of parallel computational processes so as to maintain some desired sequential relationship between programmed actions within the processes.
Event -	The occurrence of some programmed action within a process which can affect another process.
Interrupt -	An occurrence which is not a programmed action within any computational process, but which requires some response. (This is usually a signal from some external hardware device.
Task -	In the Multics sense, a virtual processor. (A single processor may be concurrently simulating many virtual processors.)
Semaphone -	A task synchronization mechanism. (first defined by Dijkstra).

Data - Information that is operated on during program execution.

Constant - A data item which takes as name its value (hence its value is fixed during program execution).

Variable - A data item which can take as name something other than its value.

Data Element - A scalar, array, or structure.

Multics - The time-sharing system developed at project MAC.

Recursion - Is the property which allows a callable program to call itself.

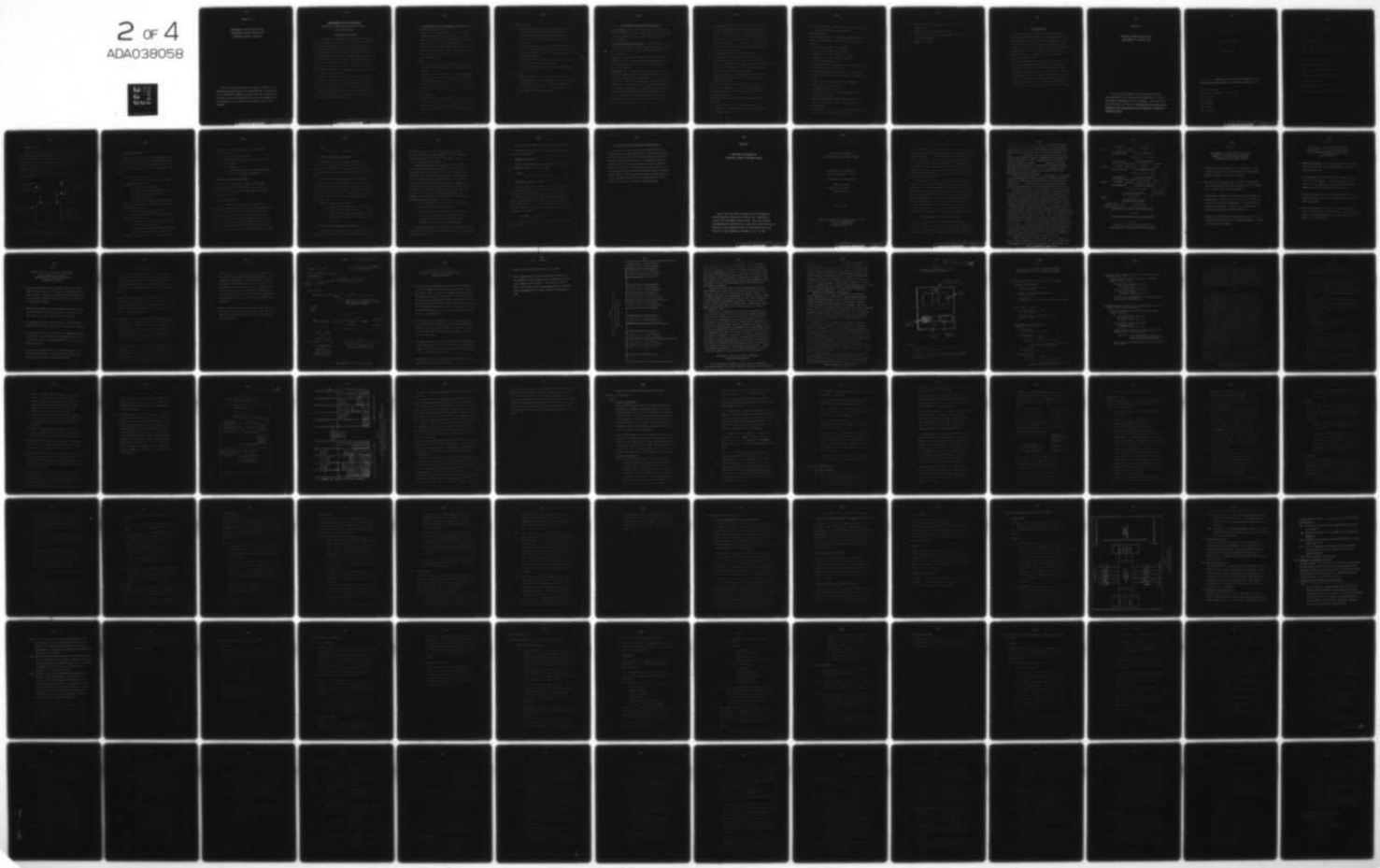
Reentrancy - Is the property which allows a callable program to be called and executed before it has completed the execution from a previous call. The results of the previous call are not affected.

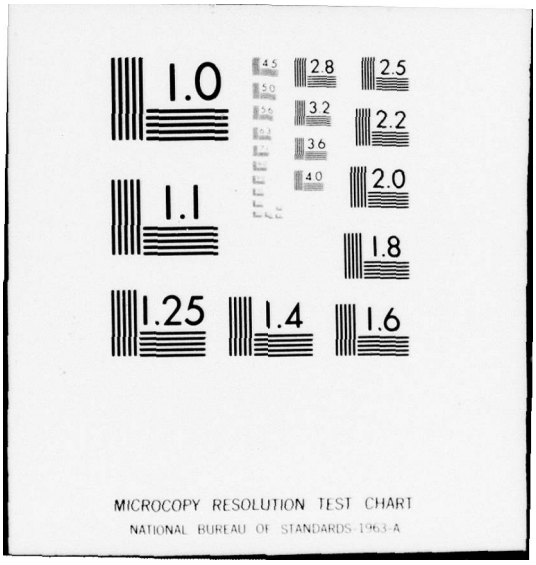
Automatic Storage - Allocation is the ability of the compiler and for the operating system to allocate temporary storage locations for designated variables without explicit instructions in the source code.

AD-A038 058 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIO--ETC(U)
JAN 77 N00014-76-C-0732

UNCLASSIFIED NL

2 OF 4
ADA038058




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SECTION III

REQUIREMENTS FOR DATA COLLECTION, COMPUTERIZED SYSTEM CHECK-OUT AND SEQUENCING CONTROL FUNCTIONS

One of the topics discussed at length at the First and Organizational Meeting of the Purdue Workshop on Standardization of Industrial Computer Languages was that listed above. The group at that time produced the attached document which was published as pp. 59-66 of the Minutes of that First Workshop.

PRECEDING PAGE BLANK-NOT FILMED

REQUIREMENTS FOR DATA COLLECTION, COMPUTERIZED SYSTEM CHECK-OUT AND SEQUENCING CONTROL FUNCTIONS

I. SYSTEM ERROR CHECKING

In the process of implementing a high-level language to free the engineer from the details of machine and assembly language programming, we are also removing a portion of the skill presently used to implement and maintain computer systems. To replace these skills, diagnostic procedures to isolate and analyze system failures, both equipment and programming, must be implemented with the language. Under these procedures, system errors should be reported to the engineer in his language; communicating the nature of the error, where it has occurred, and specific information to aid in the diagnosis and correction of the error.

Further to the definition of an error, recovery procedures are necessary to maintain the integrity of the system to the fullest possible extent. These would typically include the provisions to transfer the function (in error) to an alternate (if available), temporarily delete the function from the system, or any similar actions possible to allow the remainder of the system to continue operating.

PRECEDING PAGE BLANK-NOT FILMED

II. REQUIREMENTS FOR DATA COLLECTION - ANALOG INPUTS

1. Conversion to Engineering Units

   A. User will have the ability to specify conversion equations beyond those provided with the system.

   B. User will have the ability to change any of the parameters in the conversion equation on-line.

   B. The data can be made available to the user in either fixed or floating point format on a system basis.

   D. The data can be made available to the user in engineering units.

2. Timing

   A. User will have the ability to do either continuous scanning, random scanning or both continuous and random scanning.

   B. User will have the ability to define a sequential relationship between the variables being scanned.

   C. User will have the ability to specify the scan frequency on a point basis.

   D. User will have the ability to add or delete points on-line.

3. Digital Filtering

   A. Digital filtering will be optional on a point basis.

   B. User will have the ability to specify filtering functions beyond those provided by the vendor.

   C. User will have the ability to change all parameters on-line.

4. Limit Checking

   A. The user will have the ability to specify either one or two sets of alarm limits on a system basis.

   B. A dead band will be provided at the alarm limits and the size of the deadband can be specified on a point basis.

   C. A significant change test will be provided and the size of the change can be specified on a point basis.

   D. The user will have the ability to change any of the parameters on-line.

   E. The user will have the ability to determine the limit checking frequency on a point basis.

   F. A linkage will be provided to other system programs on limit violations.

5. Scanning

   A. The ability will be available to read a point by providing the necessary point identification parameters and other applicable scanner hardware parameters.

   B. A linkage will be provided to other system programs when scanner errors are detected.

III. REQUIREMENTS FOR A SEQUENCING LANGUAGE

1. File Structure - Due to the many programs or recipes and associated data necessary in batch processing, extra effort should be applied to the development of an efficient file structure.
2. Logical and Relational Functions - Due to the many on-off tasks associated with sequence control or batch processing, efficient and useable logical and relational functions are necessary.
3. Timing - The sequence programs used in many manufacturing processes require a rapid response including both recognition and action, to either cyclic scan programs or process interrupts.
4. Useability - It is quite important, especially for batch processing type applications, that the process engineer be able to accomplish the programming. It is his process knowledge, especially his knowledge of what can go wrong and what to do about it, which is being transferred into the computer system. In cases where it is necessary for computer oriented engineers to develop the programs, it is still necessary for the process engineer to review those programs in detail. Therefore, a self explanatory system is quite important.

A. The Application Engineer should definitely not have to concern himself with bits or byte handling, the real time operating system, interrupt responses, or routine I/O. The sequencing language should automatically make use of these to carry out its functions.

B. As much as possible, the Application Engineer should be able to use ordinary English to define his equipment and write his programs.

C. The Language should be self-documenting.

D. The system for assigning names to equipment, analog sensors, DDC loops, etc. and for parameterizing the data acquisition and DDC programs should be integrated with the sequencing language, so that names and addresses need not be specified more than once.

E. The user should be thoroughly grounded in logic and Industrial Application concepts in order to use any sequencing language.

5. That a large number of process-oriented compiler diagnostics should be provided, concerning the improper use of equipment or DDC Loops, or general specification, or programming errors.

That these diagnostics inform the programmer of such things as:

5.1 You have tried to close a non-existent valve.

5.2 You have illegally connected a DDC loop output to a solenoid valve.

5.3 You have used the same analog sensor name twice, or the same input channel number twice.

And many others.

6. The following additional functions have been recognized as being necessary or at least desirable for sequence control:

A. Turn on a two state device (name)

B. Turn off a two state device (name)

C. Change state of a two state device (name)

D. Pulse out (direction, magnitude, name)

E. Position (magnitude, name)

F. Activate loop (initial position, set point, delay time)

G. Deactivate loop (final position)

H. Function Generator (function no., cut off point)

I. Change control loop parameter (loop no., parameter, value)

J. Delay

K. Change scan frequency (point no., frequency)

L. Demand analog scan (point no.)

M. Demand digital scan (point no.)

N. Set alarm limits (point no., high/low, value)

O. Arithmetic capability

P. Initiate man-machine communication (function no.)

Q. Specify and terminate repetitive logs (log no. and time interval)

R. Specify alarm conditions for possible action - either digital or analog actuated (alarm list no.)

S. Capability to confirm results of digital and analog control actions.

T. Capability to retry a digital action.

U. Trace and snapshot requests available for diagnostic and debugging purposes.

V. Subfunction calls.

IV. RECOMMENDATIONS

1. A committee should be established which would define standard functional requirements of programming packages for such areas as data collection and man-machine communication. Other areas could include Digital scan, control, logging, alarm action and computer to computer communication.
2. Error diagnostics should at this point continue to be a functional area or a committee at the standardized software workshop meetings.
3. The evidence available indicates that at present neither of the two languages being discussed, the procedural language and the format defined language, will satisfy the functional requirements of sequence control. It is recommended that the functional requirements of sequence control be considered by a committee which concerns itself with the general question of a problem oriented language for process control, rather than a format defined language.

SECTION IV

FUNCTIONAL NEEDS FOR REAL-TIME ENHANCEMENTS TO MINIMAL BASIC

One of the first tasks of the Industrial Real-Time BASIC Committee was to develop the attached list of specific functional requirements for that language. This report was published as pp. 197-205 of the Minutes of the Second Annual Meeting of the International Purdue Workshop on Industrial Computer Systems.

FUNCTIONAL NEEDS

FOR

REAL-TIME ENHANCEMENTS

TO

MINIMAL BASIC

Prepared by the Industrial Real-Time BASIC Committee of the International Purdue Workshop on Industrial Computer Systems

Van Diehl, Chairman

Horst Halling, Chairman, European Group

Thomas M. Bell

Philip Stein

John Herbster

Dave Herman

George Janjecic

PRECEDING PAGE BLANK-NOT FILMED

TABLE OF CONTENTS

1. Introduction

2. Process Input-output

   2.1. Analog Input-output

   2.2. Digital (discrete) Input-output

   2.3. Pulse (train) Input-output

3. Access to System Time and Date.

4. Multiple Character-devices

5. Extension of Control Statement

6. Extension of Data Types and Operations

7. External Subroutines

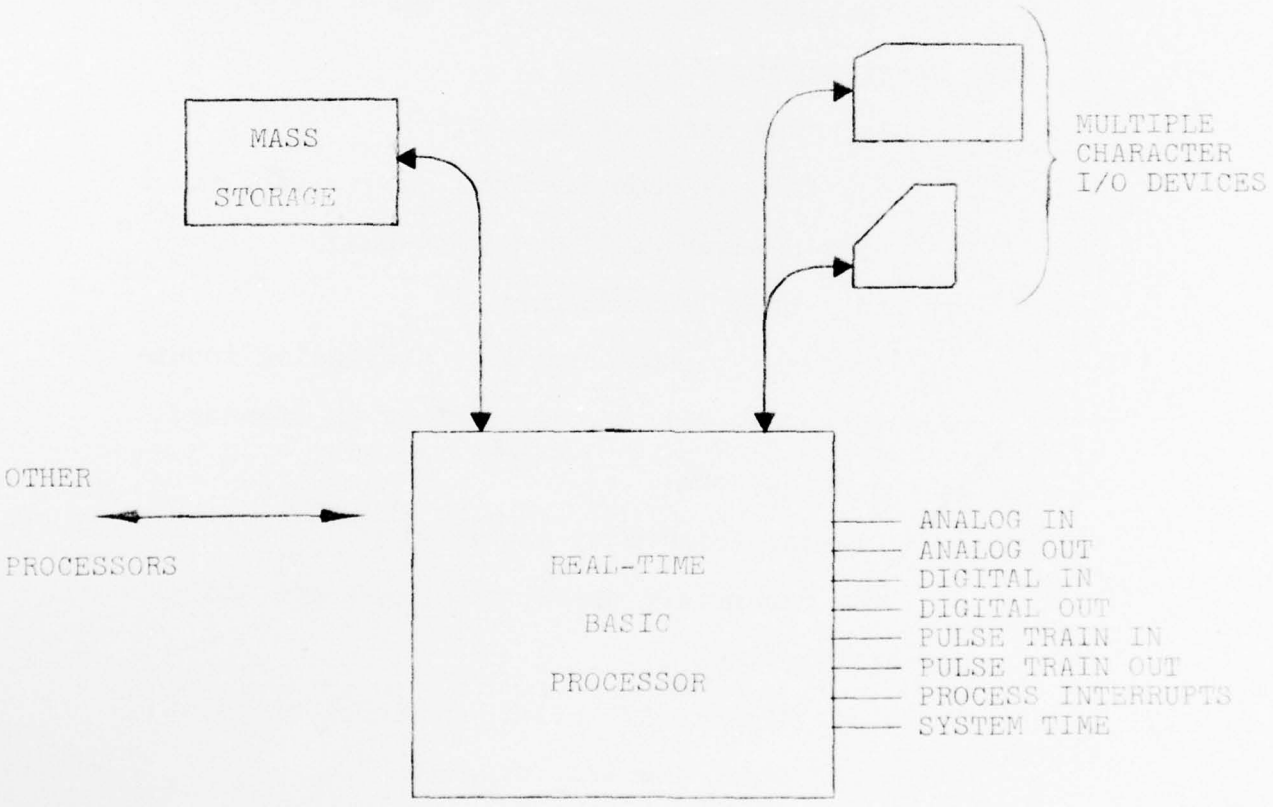
8. Extension to Error Handling

9. File Handling and Interprocessor Communication

1. Introduction

The BASIC features which have made the language successful in different application areas also apply for a class of real-time applications. A great number of implementations already exist that use various extensions to meet real-time needs. This document is an attempt to define the common requirements of the real-time user.

The following diagram is an illustration of a possible real-time system:



Syntax examples used in this document are only for illustration and are not part of the proposal.

2. Process Input-Output

Process input-output functional interfaces allow the access to specified analog and digital inputs and outputs. All the process input-output operations are executed in direct response to the corresponding statement.

2.1. Analog input-output

The parameters needed for these operations are specifications of:

(a) The hardware channel(s)

(b) Access Mode (random, sequential, etc.)

(c) Hardware-software conversion

(d) Error information

The data transferred over the analog input-output interface will be represented in standard floating point form.

2.2. Digital (discrete) input-output

The parameters needed for these operations are specifications of:

(a) Digital I/O point or series of points (discrete I/O channels)

(b) Hardware-software conversion information

(c) Error identification information

The data transferred may be to or from a standard

floating point variable or a bit string variable (see Section 6).

2.3. Pulse (train) input-output

The parameters required for these operations are specifications of:

(a) Channel(s)

(b) Hardware-software conversion information

(c) Error identification information

3. Access to System Time and Data

The current time must be available to Real-time BASIC programs for use in calculations and logging. It must give date (year, month, day) and time information (hour, minute, seconds).

4. Multiple Terminal I/O

Process Control configurations are usually not limited to a single system console. System configuration often includes teletypes, line printers and CRT's. An example is the need to print test results in terminal printers local to tests being performed. There is therefore a need to extend the PRINT and INPUT statements to include the specification of the target devices. Operator and device errors in response to INPUT statements shouldn't be fatal.

Example:

Print #6 ; < variable list >

5. Extension of Control Statements

Minimal BASIC includes control statements that sllow for the interruption of the normal sequence of execution of statements by causing execution of a specified line, rather than at the one with the next higher line number.

Real-time BASIC requires extension of the control statements to allow for the causing of the execution of a specified line number by an elapsed time having occurred, at a specific time of day, upon the occurence of an external event or error condition. Statements that enable or disable the action of such control statements are also a desirable feature.

Examples of these extensions are:

(1) Execute "line number" after 5 second delay.

(2) Execute "line number" at 10 aw.

(3) Execute "line number" when contact 5 closes.

(4) Execute "line number" if any terminal key is struck.

6. Extensions to Data Types and Operations

The number of data types in Real-time BASIC should be kept to a minimum. In the proposed "Minimal BASIC",

two data types exist. The data type floating point variable and the data type character. The corresponding data structures built of these data types are array (1 or 2 dimensional) and string.

Manipulation of analog and digital I/O data can be done using the standard floating point representation. Approprate conversion routines in analog and digital I/O statements will convert discrete (digital) inputs-outputs into the internal machine representation. In this form, no integer representation will be required. However, a bit string data structure is strongly desirable, especially in applications in automatic digital testing where large bit strings, in the order of hundreds, are required. In summary, the following process data have to have internal representation in the BASIC system:

(1) Analog input-output - Values can be represented by standard floating point variable.

(2) Digital input/output - The bit pattern of physical "0" or "1" combination can be represented by a string of "0" or "1" in a bit string or by an equivalent floating point representation.

Additional operations are required to manipulate internal representation of external discrete process data. The Boolean operators needed are:

AND, OR, NOT, exclusive OR, SHIFT, BIT TEST/SET/CLEAR

These operations should be defined as functions returning a bit string result.

7. External Subroutines

Calls to user-generated subroutines that are external to the BASIC system should be provided.

Example:

400 CALL HELP (org_1, org_2, ...)

8. Extension to Error Handling

A very important consideration in the use of real-time BASIC in industrial applications is the type of error handling capability that should be provided in the language.

Whenever errors occur during BASIC program execution (such as overflow, incorrect data input, I/O errors, etc), they should result in the setting of an error flag. Such an error flag may be tested by an extension of the control statement of the form:

5 ON ERROR GOTO (GOSUB) 100

A scheme will be provided to allow the user to identify the error cause.

9. File Handling and Interprocessor Communication

There are strong requirements for file handling in measurement and control applications. These are similar to the ones that are being considered by the X3J2 Application Oriented Committee on Science. It is important that the standard syntax for file manipulation could be extended to access files that may be resident on the mass storage of a remote computer in an distributed system network. The same remarks apply to chaining, where program chains may be resident in the same computer mass storage device or in the central computer in a distributed system network.

SECTION V

FUNCTIONAL GUIDELINES FOR INDUSTRIAL CONTROL COMPUTER SYSTEMS

One of the very early documents of the ISA Computer Control Workshop developed in 1963-64 was a preliminary version of the document listed above. This was reviewed and expanded for consideration at the First Purdue University Meeting of that Workshop in May 1972 and published in the Minutes of that Workshop as Appendix II, pp. 33-189.

PRECEDING PAGE BLANK-NOT FILMED

FUNCTIONAL GUIDELINES FOR INDUSTRIAL CONTROL COMPUTER SYSTEMS

A Proposal for Consideration

at the 1972 Meeting of the

ISA COMPUTER CONTROL WORKSHOP

Purdue University

Lafayette, Indiana 47907

May 22-24, 1972

December 1971

Purdue Laboratory for Applied Industrial Control
Schools of Engineering
Purdue University
Lafayette, Indiana 47907

PRECEDING PAGE BLANK-NOT FILMED

INTRODUCTION

At the April 22-23, 1971 meeting of the Computer Control Workshop (succession to the Workshop on Direct Digital Computer Control (13)) it was suggested that sufficient time had passed so that the Guidelines and Questions and Answers document of the original Workshop required modification and updating. The facilities of Purdue University were offered for holding the required new Workshop and the date of May 9-12, 1972 was proposed. The attached proposal will be used as the initial discussion point for this new Workshop.

In developing the overall system to be proposed here a major if not overriding consideration must be the reduction of personnel requirements to implement an installation in terms of the original engineering layout, the mathematical modelling and control strategy development, the coding in an appropriate programming language, and finally the checkout and testing of the final installation. Other important considerations are developing the maximum overall reliability of the resulting system, the minimum total cost at final commissioning, as well as a maximum flexibility and ease of making post-commissioning changes.

TASK ASSIGNMENTS AND HARDWARE FEATURES

Figure 1 outlines the hierarchy organization which comprises the largest scale version of our proposed standard process control computer system. Level One and Level Two machines are very similar in function-differing only in the extent of the

PRECEDING PAGE BLANK-NOT FILMED

control functions assigned to each. The dedicated digital controller of Level One handles complex devices, including chemical analyzers such as chromatographs, and specialized feedforward and noninteracting - multivariable control set-ups. The direct digital controller of Level Two handles a much larger number of three-mode and related type control loops(12). It also communicates with the plant operator through a console, composed of cathode ray tubes and a keyboard. Communication between all digital computers and with all consoles is by digital signals. Connections to analog signals are required only of the first and second level machines.

Table I specifies some of the assignments of tasks within the hierarchy which are necessary to achieve the simplification of system and function which it is possible to develop here. Table II presents some of the desirable features of the machines to be used themselves while Table III does likewise for the other elements of the total system.

In order to achieve our main object of reducing overall systems cost and of simplifying greatly the engineering, installation, and commissioning of an industrial control system, it is necessary to modify greatly the previous organization of the data collection and conversion and the control correction output distribution parts of the system. In place of the previously common methods of using individual pairs of leads between sensor and central processer and between processor and final actuator we wish to adopt the concepts recently perfected by the aerospace industry. This is to use a set of remote multiplexers and a single "data highway" carrying only digital signals between these multiplexers and the computer itself (6). Such a concept has been technically possible for sometime (7). However, only recently has integrated circuitry made remote multiplexers and multiple A/D converters economically practical and computer systems fast enough to handle the data load necessary for the method of operation of the system which is to be proposed here (2,3,5).

Figure 2 diagrams the data collection and conversion part of the system under discussion. Table IV outlines our requirements for its design and operation considering such

(Text continues on page -125-)

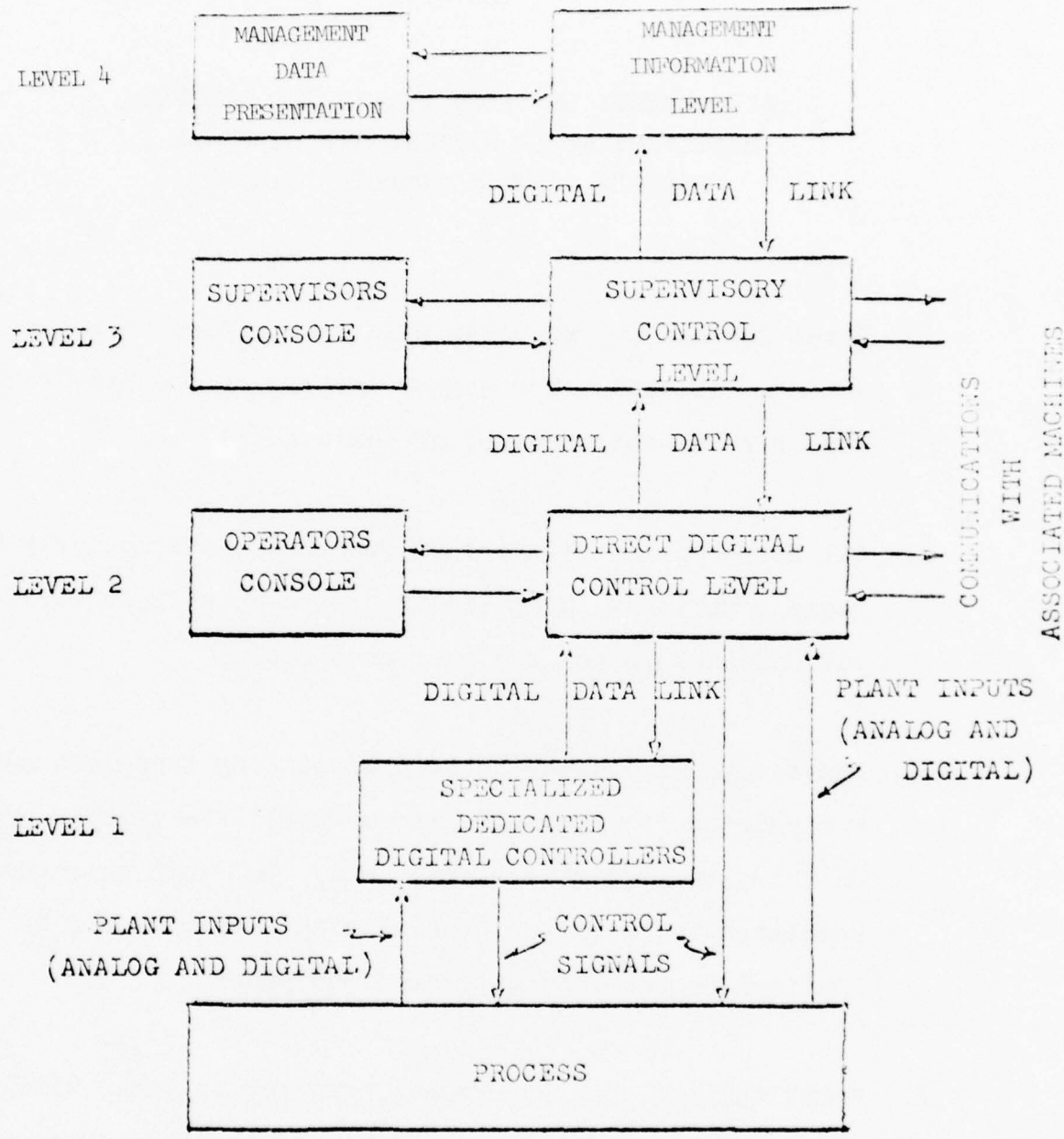


FIGURE I

OUTLINE OF HIERARCHY ORGANIZATION FOR

STANDARDIZED PROCESS COMPUTER CONTROL SYSTEM

TABLE I

ASSIGNMENTS OF TASKS WITHIN THE CONTROL HIERARCHY AS IT AFFECTS THE STANDARD PROCESS CONTROL COMPUTER SYSTEM

1. Level One and Two Machines will be confined to tasks of data collection for higher members of the hierarchy and/or to dynamic control of the plants.

2. All large scale optimization functions, particularly those comprising large linear programs, will be carried out on Level Three or higher machines.

3. Non-control hierarchy related accounting functions and engineering or other off-line work will be carried out on third or higher level machines. But then only when absolutely necessary. Generally this work should be confined to separate unrelated equipment.

4. Compiling and related program preparation tasks will be carried out on Third Level or higher machines or on unrelated off-line equipments.

TABLE II

MINIMUM REQUIRED CHARACTERISTICS AND CAPABILITIES OF COMPUTER MAINFRAMES USED IN THE PROPOSED STANDARD PROCESS CONTROL COMPUTER SYSTEM

1. Lower level machines in the hierarchy should have only internal read-write or read-only memories to assure maximum reliability.

2. All mainframes should have parallel organization and state-of-the-art speeds. They should also have sufficient built-in hardware facilities to make their programming by off-line means readily feasible.

3. Interface equipment must be designed and built to the same quality and reliability specifications as for computer mainframes.

4. Maximum reliability considerations must be handled by fault tolerant designs, dual systems, or established fail-upward, backup arrangements for all machines carrying out dynamic control functions.

TABLE III

SOME PROVISIONS OF SYSTEM (NON-COMPUTER)
HARDWARE USED FOR STANDARD
PROCESS CONTROL COMPUTER SYSTEM FOR
INDUSTRIAL COMPLEXES

1. Single sensors and single, twisted, stranded, shielded pairs of lead wires shall be used for each analog point read. Where redundancy is required for safety this will be by entirely separate sensors and sets of lead wires to the process location.

2. Remote multiplexers will be distributed about the plant in such a manner that the longest sets of analog lead wires shall be in the order of 100 feet or less.

3. All switches shall be solid state and will be designed such that all failures will cause a "fail open" condition.

4. All switches shall be dual giving two separate data paths through them. Each arm should be separately testable by an external signal to permit determination of a possible failure condition.

5. Each remote multiplexer shall be equipped with its own A/D and D/A converters to permit digital communication throughout the remainder of the data system.

TABLE III (CONT.)

6. All remote multiplexers attached to one direct digital control center shall be connected to their corresponding computer systems by a single data path consisting of three coaxial cables; one for data in ; one for data out; and one for timing, inquiries and other operational signals.

7. Redundant signal paths may be supplied if reliability requirements so dictate. Such paths should be routed spaced sufficiently far from the primary path that no single accident, natural disaster, or act of minor sabotage can sever both paths.

8. System power shall be supplied from a backup power system such as an amplidyne plus gas engine driven motor generator, by a battery system floating on line or by another equally capable system. Such backup must be capable of keeping the system operating without interruption or without causing system error for one hour or twice the mean repair time, whichever is longer.

9. All maintenance shall be carried out under a philosophy of replacement of the compent with a spare with subsequent fault analysis and repair off-line. Thus spares must be maintained of all items of the system which can be considered minimum entities and whose fault causing

TABLE III (CONT.)

failure can be so identified as to unit involved and thus readily pinpointed. Appropriate levels might be complete chromatographic instruments; mini-computer mainframes; computer memory modules if specific; A/D converters; multiplexer modules; etc. Use of manufacturer supplied spares and maintenance personnal may have to be involved in such work because of proprietary or commercial interests or contractural arrangements arrived at.

10. Single output transducers and single output units shall be fused for each control point unless very stringent reliability requirements dictate the use of multiple control elements.

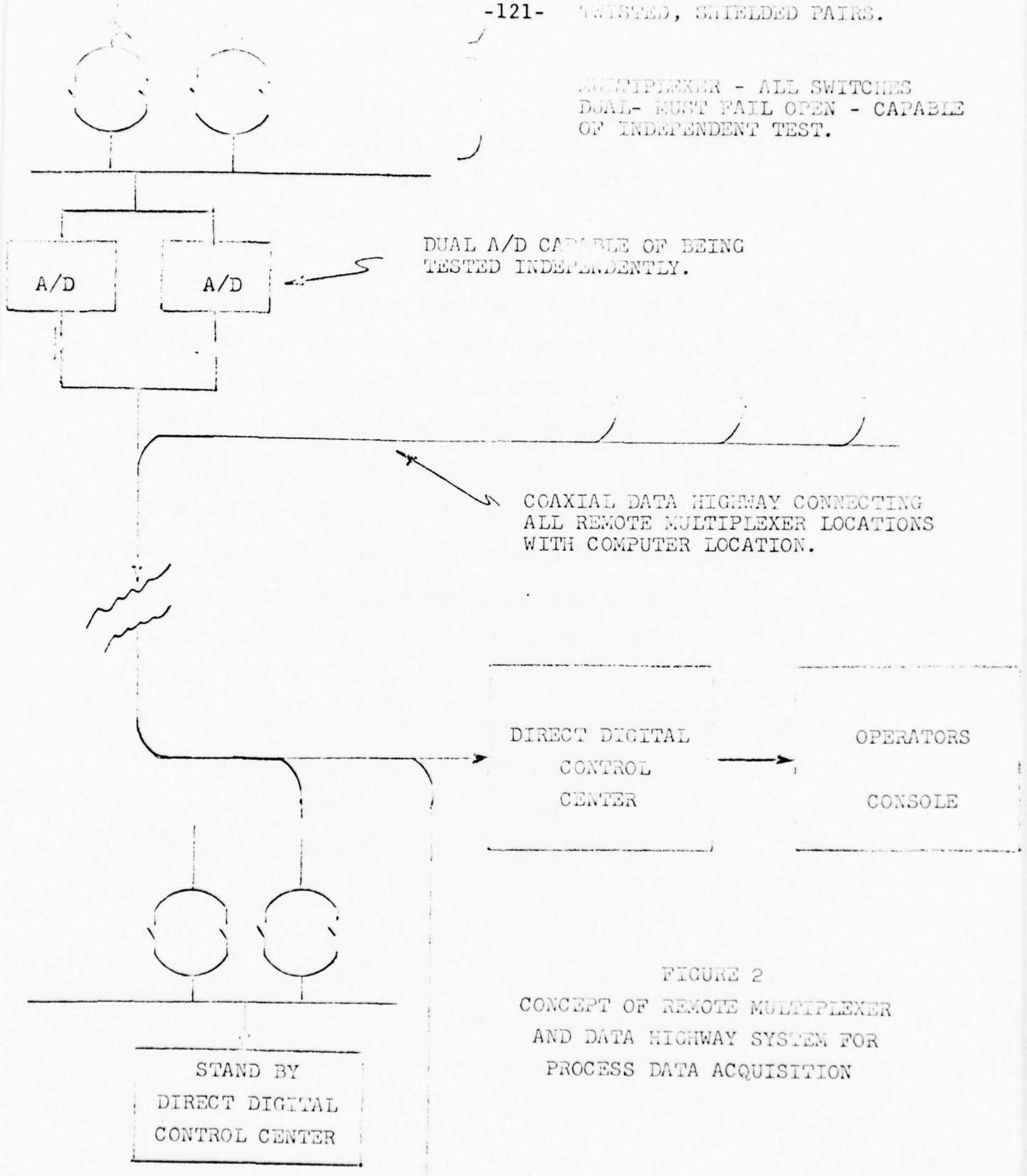


FIGURE 2
CONCEPT OF REMOTE MULTIPLEXER
AND DATA HIGHWAY SYSTEM FOR
PROCESS DATA ACQUISITION

TABLE IV

REQUIREMENTS ON OPERATION OF
PRODUCTION CONTROL COMPUTER SYSTEMS FOR AN
INDUSTRIAL COMPLEX

1. System will be operated in such a way that a redundant or a fail-upward backup of any component may be carried out easily. This can be best accomplished by polling each separate point on each separate remote multiplexer in turn on each Direct Digital Control Center until every one in each center has been polled singly in its turn, ie., all on Remote Multiplexer One of Center One, all on Remote Multiplexer Two, etc., then all on Center Two, and so forth. (See Figure 3)

2. All data collection computers will be equipped with direct memory access devices to permit storage directly in memory of the last sensor reading on each point. This is to avoid overloading the main arithmetic of machine with routine tasks.

3. The data acquisition system will operate on a one-tenth second cycle, ie., every point is to be read each tenth second. This will allow following even the fastest variables of an industrial process.

4. Analog data will be converted to a resolution of thirteen bits plus sign (14 bits total).

TABLE IV (CONT.)

5. Digital data will be handled as 14 bit words.

6. Limit checking will be done on entry into memory if the computer can be equipped with a comparator on the direct memory access device. Otherwise it will be done as the first item of data manipulation by the computer in preparing its control computations from the data.

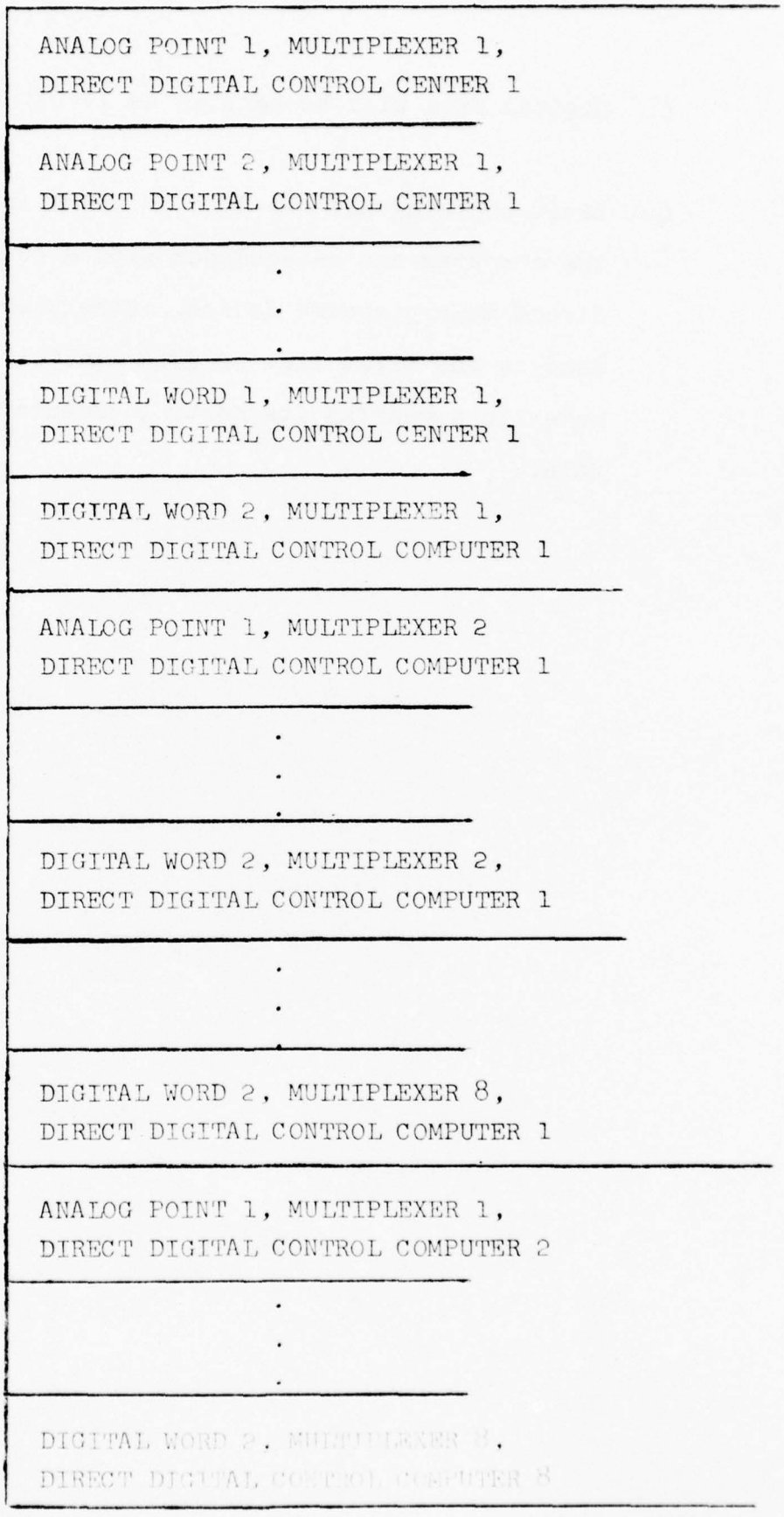


FIGURE 3
DIAGRAM OF THE POLLING SEQUENCE USED FOR ALL
POINT SAMPLING FOR CONTROL OF
LARGE INDUSTRIAL COMPLEX

topics as sampling rate, data word length, etc. In Figure 2 it should be noted that critical elements of the data path such as multiplexer switches and the A/D converter itself are dual with independent operation so that failure of any single element will not disable a particular input. We specify also that swiches should "fail open" by design (in contrast to the usual operation of reed relays) and that these elements should be capable of being tested independently during preventive maintenance procedures.

Reliability at the Level 1 and 2 computer ends of the data requisition system can be achieved by the use of dual computers, one in a use, the other in a standby status. For a large system such as contemplated by Figure 1 a single standby computer might be used as backup for all Level 2 machines of the complex as diagrammed by Figure 2. Again the recent drastic price reductions of capable small computer mainframes makes economically practical a concept which is not new in the industry (4,10).

Another major concept of this proposal needed to help achieve system simplification is that all operator communications will be through a color cathode ray tube (CRT) presentation of both alphanumeric and graphical data (9,11,13). All operator actions will then be conducted through keyboard entry from this same small console containing the CRTs. Dual CRTs will be specified with independent and interchangable functioning for reliability reasons. No hard copy (printed) output is contemplated at Level 1 on Level 2 stations. All of this latter can be handled at Level 3 or 4 locations when needed for accounting or historical records. Acceptance of this part of the proposal would eliminate the massive central control rooms now used by all process industry plants with their attendant major design, installation and check-out difficulties. Figure 4 sketches one concept of the possible appearance of such an operator's station for a Direct Digital Control Center.

AN EXAMPLE SYSTEM FOR A LARGE INDUSTRIAL MANUFACTURING COMPLEX

If the industrial computer control system contemplated here were assigned to a large petroleum refinery or petrochemical

complex it might be required to handle up to 1500-1800 analog inputs and an equal number of digital signals. If the conditions of the previous Tables are imposed an organization and data transmission load such as is discussed in Table V results. Note that 8 separate Direct Digital Control Centers (Level 2), each with 8 individual remote multiplexer stations are proposed as an example. The remote multiplexer stations may or may not each contain one or more Level 1 computer systems - this would be a function of the plant unit involved and its own chemical analysis and advanced control system requirements. A single Level 3 unit is envisioned here for purposes of discussion.

Please note that Table V assumes that the socalled "data highway" will be composed of three separate coaxial leads, one for data input to the Control Center, one for control correction outputs back to the process, and the third for system timing and selection signals. While one coaxial cable could probably easily handle the load contemplated here, the use of three simplifies greatly the computer programming problem without appreciable physical cost increases. Note also that the control correction and timing networks will be very nearly duplicates of the data acquisition network of Figure 2.

Plants which do not require as elaborate a system as is called for in Figure 1 can still make use of the concepts outlined here. Use of Level Three or Four machines should always imply the existance of the lower members of the hierarchy if only in a data collection, ie, noncontrol, function. The reverse case is, of course, not true since a Level Two machine can very well stand alone in an appropriate plant process control situation. The Level Three machine must have a source of plant data and this might just as well be collected by sublevel digital systems as by means of plant input fed into an interface system which is part of the Level Three machine itself.

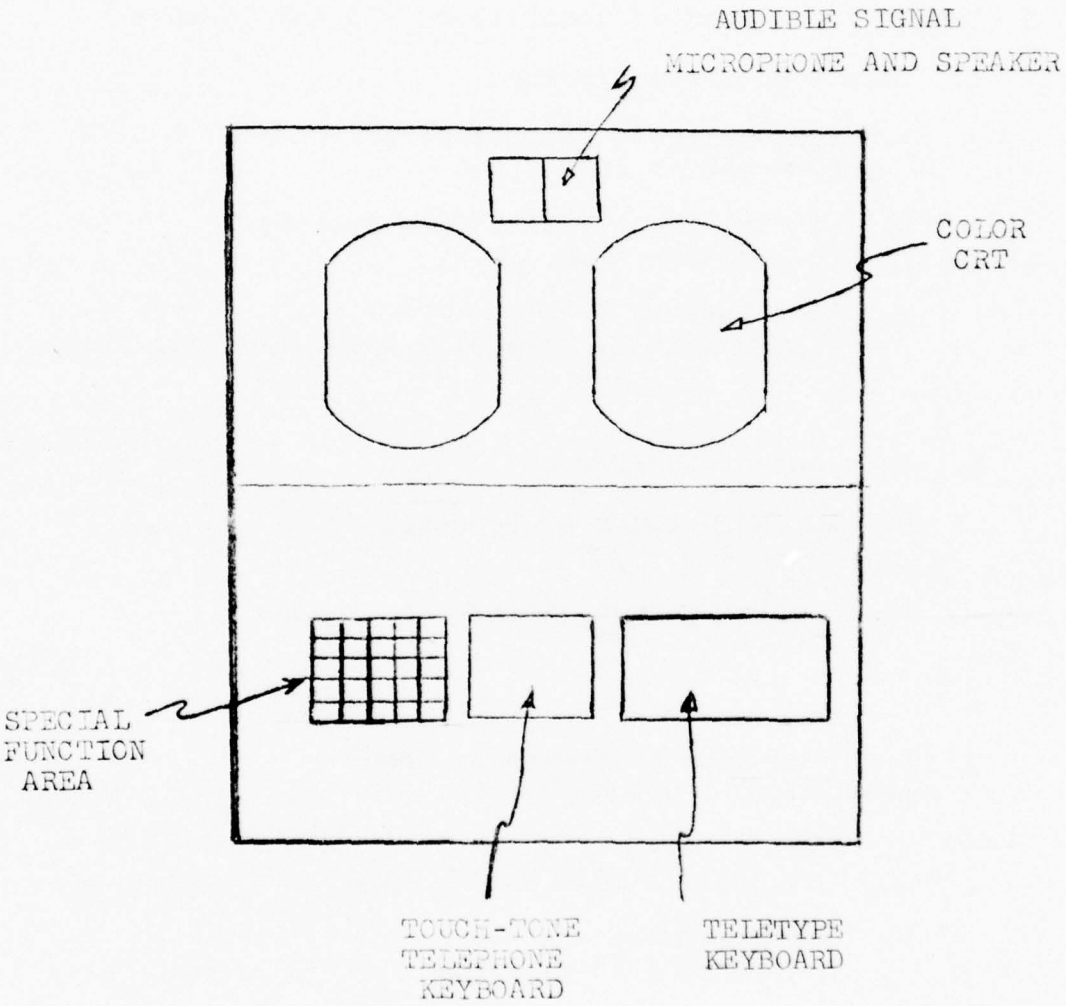
For systems which do not require a Level 3 station it may be necessary to specify a hard copy device at the Level 2 location to provide the accounting and historical records which are necessary for the proper operation of the plant.

(Text continues on page -130-)

BEST AVAILABLE COPY

FIGURE 4

SOME FEATURES OF OPERATOR'S CONSOLES FOR THE PROPOSED PROCESS CONTROL SYSTEM



FEATURES

1. ALL OPERATOR COMMUNICATION BY CRT - NO HARD COPY PRINTED MATERIAL CONTEMPLATED.

2. 2 CRT'S, NORMALLY ONE ALPHANUMERIC AND ONE GRAPHCAI. INTERCHANGABLE IN CASE OF FAILURE OF ONE AND WHILE AWAITING REPAIR.

TABLE V

DATA LOAD PER PRODUCTION CONTROL COMPUTER SYSTEM (LEVEL 3) FOR AN INDUSTRIAL COMPLEX

I Total System Load if Centralized for any Purpose

1. Data Input from Process

   a. Analog Inputs from Process
   (8 X 240) = 1920

   b. Digital Input Points from Process
   (8 X 224) = 1792

   c. Data Rate
   [(1920 X 14) + (1792)]10 = 286,720 bits/second

2. Data Output to Process

   a. Analog Outputs to Process
   (8 X 120) = 960

   b. Digital Output Points to Process
   (8 X 112) = 896

3. Required Data Transmission Load
   Assuming 0.1 Second Cycle

   a. Analog Input
   (1920 X 14)= 26,880 bits

   b. Digital Input
   (1792 X 1) = 1,792

   Total Input 28,672 bits per cycle

   or 286,720 bits/second input with polled burst mode operation

   c. Analog Output
   (960 X 14) = 13,440

   d. Digital Output
   (896 X 1) = 896

   Total Output 14,336 bits per cycle

   or 143,360 bits/second output with polled burst mode operation.

TABLE V (CONT.)

II Direct Digital Control Centers per Productor Control System Center (Level 1) -

8 Direct Digital Control Centers per Production Complex Control System

256 Total Data Words Input Each
240 analog points
224 digital points (16 X 14)

128 Total Data Words Output Each
120 analog output
112 digital output (8 X 14)

Plus all local requirements for system operation and operator communication.

III Remote Multiplexer Stations per Direct Digital Control Center (Level 1 and at Process) -

8 Remote Multiplexer Stations per Direct Digital Control Center

32 Total Data Words Input Each
30 analog points
28 digital points (2 X 14)

16 Total Data Words Output Each
15 analog output
14 digital output (1 X 14)

A/D Converter Conversion Rate Assuming Polled Operation and Burst Mode Becomes

2048 X 10 = 20,480 equivalent conversions per second allowing equal polling time for digital point words

Required A/D Conversion is 14 bits (13 bits data plus sign)

As mentioned earlier, complete reliability of all dynamic control functions is an absolute necessity. This can be achieved by fault tolerant design techniques, by redundancy of the equipment used itself such as master-slave systems, or by allowing higher members of the hierarchy to take over the functions of lower members which have failed. The interests of simplifying the programming task would favor the second method.

PROGRAMMING

In order to achieve our overall system objectives as regards programming the items listed in Table VI are necessary. Fortunately a major advance in this direction is already underway with the results to date of the Workshop on Standardization of Industrial Computer Languages (1). This group has proposed that the standard compiler level language be ANSI Standard FORTRAN augmented by a set of standardized CALL statements to allow the additional functions necessary for process control. Another key feature of our proposal here is the elimination of compiling and other programming features from the presumably small First and Second Level machines. This work would be carried out on higher level members of the hierarchy or other larger off-line machines (12).

It should be specifically noted at this point that no requirements are placed on standardization of machine speeds or hardware characteristics. Any computer may be used for any of the tasks involved here so long as its reliability, flexibility, speed, and cost characteristics match the task desired and - most important - the manufacturer has provided the executive programming and off-line program generation capabilities called for here.

It should also be noted that the program standardization desired also achieves another very important gain for the user industries - that of program transportability. That is, a program written in one or other of the higher level languages specified here can be run on any machine whose vendor has equipped it with the necessary programmability called for in this proposal.

(Text continues on page -136-)

TABLE VI

PROGRAMMING REQUIREMENTS FOR ACHIEVING
A DESIRABLE STANDARD PROCESS COMPUTER CONTROL SYSTEM

1. A high level compiler type language or one of a very small number of problem oriented languages must be used for all user group conducted programming. For example, a language which includes ANSI Standard FORTRAN (X3.9-1966) as a subset admirably satisfies such a need (8).
2. The high level process control programming language should contain the necessary capabilities to permit the following industrial control functions to be programmed completely in this language:
   a) Bit Manipulation
   b) Bit Testing
   c) Process Interrupt or Priority Handling
   d) Interrupt Inhibit and Permit
   e) Parallel Task Execution
   f) File Handling
   g) External Analog and Digital Data Input and Output
3. Goal of software design should be to minimize the amount of specific information which a process engineer needs to know about a control computer and its associated software to implement computer control for a known plant.
4. Specialized process control programming systems should be developed which carry out the following:
   a) Make the organization and development of a process

TABLE VI (CONT.)

control computer system as simple as possible.

b) Contain an organizational method by which the user can carry out any functions possible with conventional control systems with only a very minimum knowledge of programming such as by using a "fill in the blanks" technique.

c) Preserve at least as much compatibility between the control practices of different industries as is possible with conventional analog control hardware.

d) Permit on-line addition or deletion of inputs, control computations, and outputs, and changes of coefficients, without recompilation or reassembly.

5. Provisions must be available to compile all languages of Item 1 above on larger computers than the object machine of the control application in question.

6. Assembly or machine language programming must be restricted to specific functions which are part of the problem oriented language features and which are carried out by the manufactures or system house producing the control equipment.

7. All component computers of the overall system except those dedicated to very special first level tasks must have sufficiently powerful executive programs to permit the ready change of parameters through operators' consoles or the action of higher level computers in the

TABLE VI (CONT.)

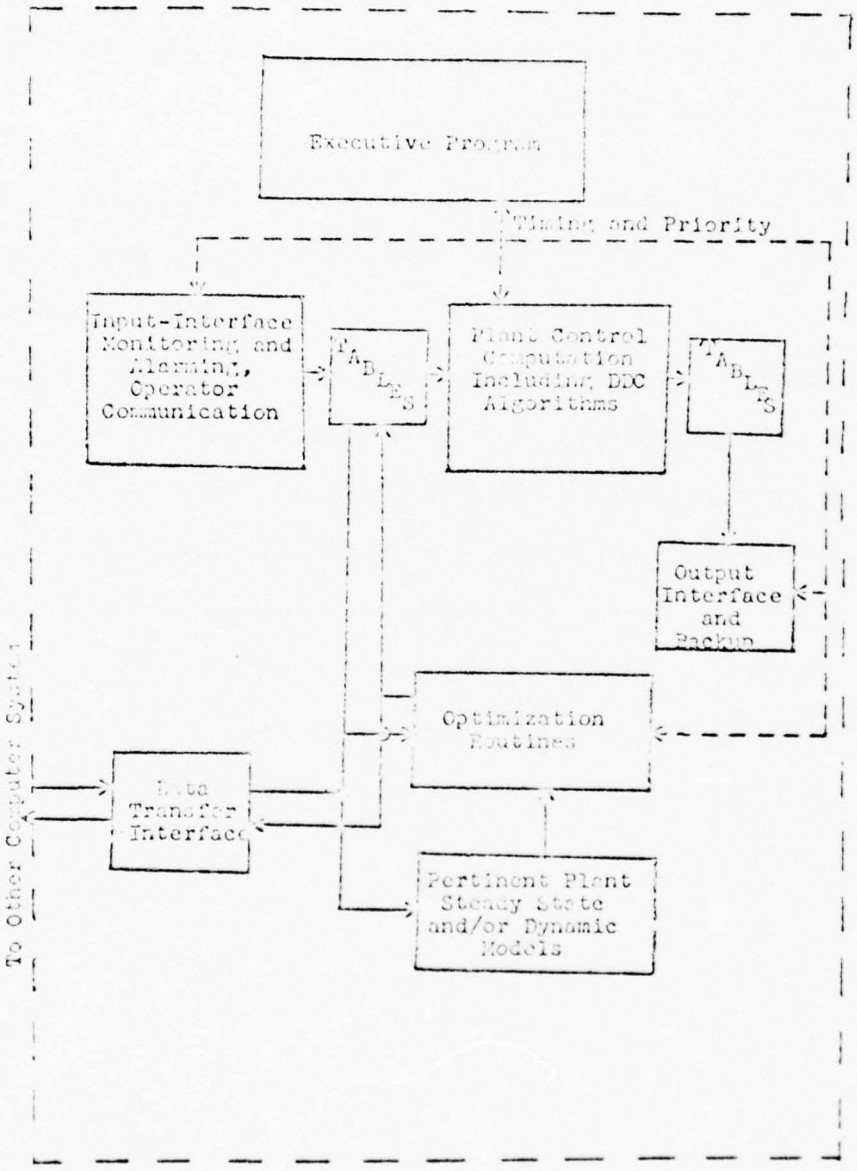
system. Provision must be made for complete and accurate documentation of all such changes. This executive should be produced by the manufacturer or systems developer.

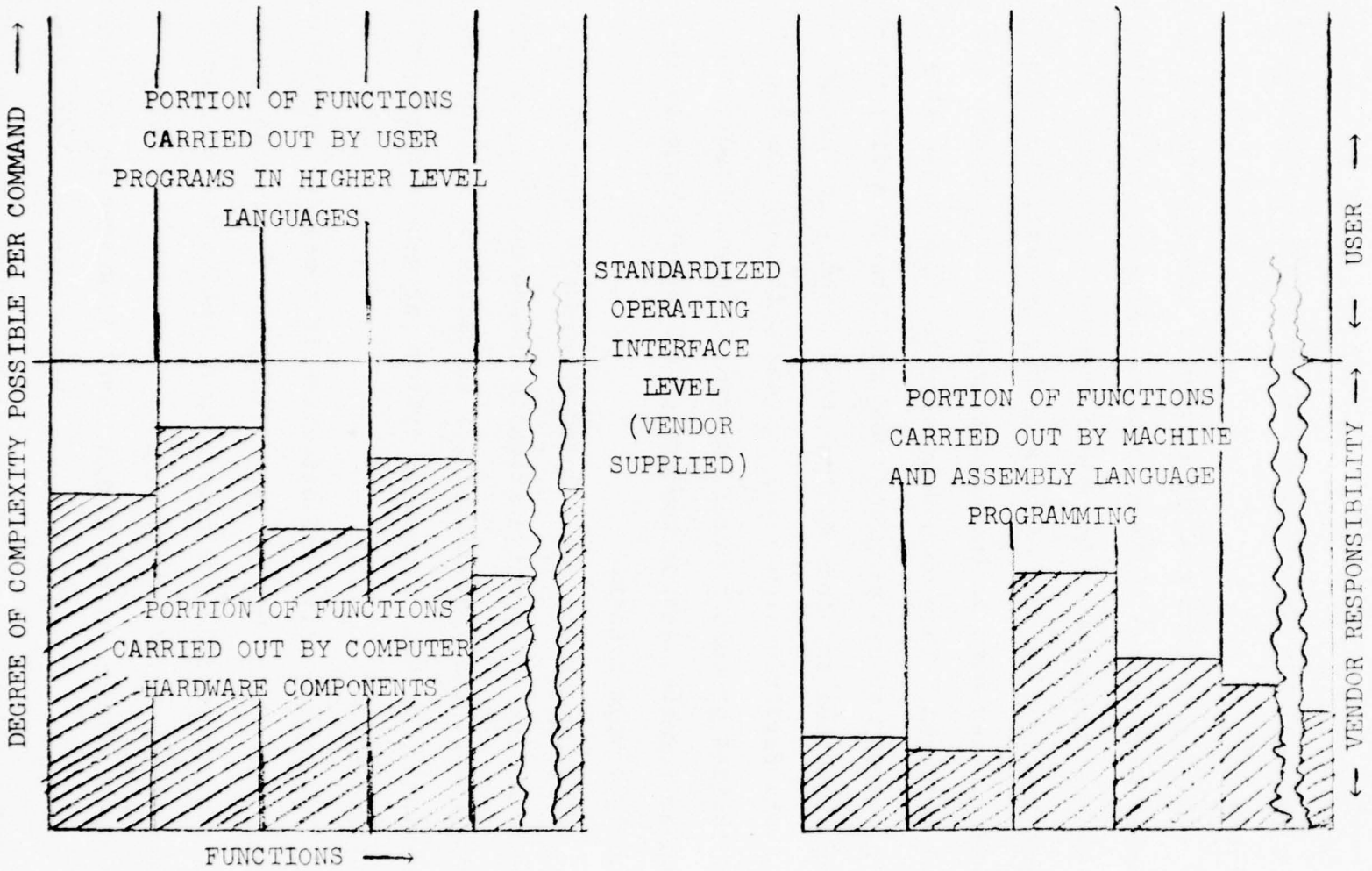
8. Software systems should be developed in a modular fashion to the maximum extent possible with the functions of the modules rigidly defined. These modules should communicate through tables to the extent possible. Table format should be such that they will permit substitution of complete blocks as newer, more efficient forms are developed and reassembly of the program without the necessity of reworking the other blocks. Figure 5 presents one form which such a system might take. This will permit standardized software to be used in a multi-computer system.

BEST AVAILABLE COPY

FIGURE 5

BLOCK DIAGRAM OF MODULAR INDUSTRIAL CONTROL PROGRAM SYSTEM





A. SOPHISTICATED COMPUTER HARDWARE

B. SIMPLE COMPUTER HARDWARE

FIGURE 6.

DIAGRAM OF RELATIONSHIP OF MACHINE HARDWARE FUNCTIONS, VENDOR SUPPLIED PROGRAMMING AND USER DEVELOPED HIGHER LEVEL LANGUAGE PROGRAMS

The goal for the standardization effort can be stated as follows (1):

"As a long term goal it should be unnecessary for user company systems and instrumentation engineers and programmers to have a knowledge of the basic machine language of the computer involved or even of an assembly language for this machine. However, they must be thoroughly grounded in computer and control concepts. They should be able to communicate with the system, construct new programs, and initiate or modify the sampling sequences and data manipulation techniques through the use of a standardized high level language, through the use of a specialized process control program which can handle simple tabular formats, and/or through an operator's console in some equally simple manner.

Programming systems adopted should also have as long range goals machine and configuration independence. They should, in addition, contain provisions for automatic or semiautomatic documentation of all changes to systems program and/or system configuration."

The nearly universal attention in these and similar efforts toward the use of higher level languages means that the vendor must be responsible for producing a combination of computer hardware and of operating system programs which will accept the user's programs written in the higher level languages in the most efficient manner. Figure 6 diagrams this situation, indicating that vendor hardware and software must supplement each other to achieve an equivalent result. A relatively simple computer

requiring a much higher use of software accomplished functions would thus be equivalent, except for speed of operating, with a much more sophisticated and efficient computer with a correspondingly smaller operating system. Present trends in small computer architecture would indicate that the Type A machine of Figure 6 would probably be the more popular in the future. This is despite the fact that most present systems tend toward Type B.

DETAILED AND SPECIFIC FUNCTIONAL GUIDELINE ITEMS

I. HARDWARE CONSIDERATIONS

1. SCOPE

1.1 General Requirements

These guideline items describe the nature, purpose, requirements, and physical equipment desired for digital computer based controllers for possible use as digital control systems for various sized processing units and/or complete plants. Such use would have the objectives of reducing capital costs of plant control systems, and of providing a greater flexibility, reliability and adaptability for plant control functions.

Since reduced capital cost is a major objective for these systems, the vendor is requested to evaluate the desirability of all convenience and flexibility features listed herein versus overall system costs. Cost reductions should be secondary only to system reliability, on-line availability, and operability in machine design considerations.

1.2 Specific Requirements

1.21 Since such computers will normally operate on a 24-hour per day, seven day per week schedule, emphasis must be placed on an extremely high "availability" of the equipment for continuous operation in process control service in normal plant control room environments. This required "availability" (See Item 3.1) should be implemented by one or more of the following techniques:

1.211 An especially high reliability of individual components and units of the computer itself.

1.212 The possibility of preventive maintenance and/or replacement of components while the computer is operating "on line".

1.213 Cabinet and circuit layout design to permit rapid and convenient troubleshooting and maintenance in event of computer failure. It should be emphasized that computer maintainability is much more important in the applications considered here than are the corresponding space requirements.

1.214 Duplication or redundancy of certain critical parts of the total computer control system with provision for automatic switch-over to standby units in the event of failure of the primary working unit.

1.215 Use of fault-tolerant, self-diagnosing or self-repairing type circuitry in the non-completely redundant units of the computer control systems.

1.22 The equipment shall be compatible for use with a backup power supply, such as batteries, which can keep the computer operating without interruption for at least one hour or twice the mean repair time, whichever is longer, in the event of a main line power failure.

1.23 The equipment described herein must have the following operating capabilities:

1.231 Read-in and alarm the measured values of a specified group of plant process variables.

1.232 Computation of the necessary corrective control action according to some established dynamic control equation or equations.

1.233 Actuation of final plant control elements such as control valves according to the results of the computation of Item 1.232.

1.234 Provision for display of selected plant variables for plant operators information where the control computer system is not directly associated with another supervisory or optimizing control computer system already equipped with suitable display facilities.

1.235 Provision for read-in of the results of manually conducted analyses and other plant tests.

2. DETAILED SYSTEM DESCRIPTION

2.1 Data-Input Equipment

2.11 The input equipment shall accept analog and/or digital pulse information from conventional industrial transducers. This information may be an electric current signal in the 1-5, 4-20, or 10-50

ma range, a voltage signal in the millivolt range or a series of contact closures.

Input equipment must be compatible with either mechanical (mercury wetted or dry reed) switching for low level signals or solid state switching for fast switching of high level signals.

If economics requires it in order to achieve desired switching speeds, a preamplifier may be used on each low level (millivolt) input prior to scanning to allow switching only of high level signals. Overall system economics (total computer system plus input) should be the deciding factor.

2.12 It is desired that all accuracy possible with present-day electronic instrumentation be retained. Therefore, analog-digital conversion system must be able to convert signals to an accuracy of at least one part in 1024 (i.e. 2^{10}). Accuracy shall be considered as the result of comparing the value of the input signal presented to the computer system data interface to the process with the number which is finally stored in computer memory.

2.13 Maintenance of the accuracy of 10 binary bits (1 part in 1024) is necessary in all computing elements to maintain the above input signal differentiation level throughout the system except as mentioned specifically below.

2.14 Sampling of analog input information from chemical and petroleum type plants shall be at such a rate that the memory location assigned to each plant input variable is updated at least as often as follows:

2.141	Approximately 60%	- Once per second
2.142	Approximately 15%	- Once per five seconds
2.143	Approximately 25%	- Once per twenty seconds

In general, flow variables will fall in Paragraph 2.141, level and pressure variables in 2.142 and temperature variables in 2.143. In addition, the following special cases apply:

2.144	Composition Variables	- Approximately once per twenty seconds or as often as supplied by detecting instrument.
2.145	Manual Inputs (Set Points, etc.)	- Approximately once per second
2.146	State Variables (Digital Inputs)	- Approximately once per second

In the above connotation, sampling shall also include the operation of determining the presence of any alarmable condition within the variable and making an appropriate indication of such condition.

Sampling of manual inputs at rates approximately once per second is necessary in order to maintain operator's sense of direct control of plant operation. Time delay in this function may hinder operator

acceptance of this method of control.

2.2 Operator's Console

To permit ready communication between the operator and the control computer system, a console is desired. It should include the following items:

2.21 Facilities for manual entry of set-point data for each control loop.

2.22 Facilities for manual adjustment of the loop alarm settings and individual gains of each control mode for each separate control loop. Since these will be adjusted by more experienced personnel and with less frequency than will set points, a time-shared or multiplexed device could readily be used for this purpose. This should be a limited access facility with key lock or similar device.

2.23 A device for read-out under operator command of values of set points, and of each mode gain for each control loop. A multiplexed or time-shared device may be used here. Either a digital or an analog type output may be considered. The particular variable or setting whose value is to be presented will be selected by a system of switches, toggles, or similar devices.

2.24 Facilities for manual movement and adjustment of the settings of the process valves or other final

plant control elements in the event of a computer outage or input transducer failure.

2.25 Alarm indication and reset devices. See below (Section 2.4) for a more complete discussion of alarm requirements.

2.26 A method of indication of the value of any or all of the plant process variables under control. Such facilities may be low accuracy (~ 5%) continuous devices (one for each variable) and/or precision (~ 1%) time shared or multiplexed devices.

2.27 Since all aspects of manual entry of data, of set point adjustment and of read-out of various variables relate to the operator's use and acceptance of direct computer control, the considerations of Item 2.14 above apply. Therefore, all functions of this type must take place within the one second period mentioned.

2.28 If at all possible in terms of economic considerations, operator functions should be carried out by means of color, cathode ray tube, read-out units. Units should have automatic character generation functions as well as full vector capability. They should have full-page buffer capability to decrease computer system overall data-transfer loadings. Scopes should be self-refreshing.

2.29 Consoles should be equipped with teletype compatible

keyboards, ten digit numeric keys, and a set of keys which can be used to initiate special function in the system.

2.3 Logger

To allow the operator to readily follow the operation of the plant unit or units being controlled by the direct digital control computer, an optional logging device may be required. Suggested features of the logger are as follows:

2.31 Logging of a number of functions equal to approximately half of the variables being controlled will be required.

2.32 Conversion of input variables to engineering units and the development of functions of several of these variables will facilitate the operator's use of such data. Thus, some computation facilities other than the control correction computation previously mentioned would be desirable in the machine.

2.4 Alarm Unit

An alarm unit should be included in the operator's console. It should have the following characteristics for maximum utility to the plant operator. An alarm for each loop being controlled is desired. The functions described may be incorporated in an operator's CRT display if desired.

2.41 It should contain one or more alarm signal lights. This light should flash whenever one of the process

variables under control reaches either the high or low limit previously established. If more than one plant unit (distillation column, reactor, etc.) is being controlled by the computer, there should be a separate signal light for each plant unit under control.

2.42 It should also include an audible alarm which would sound whenever any process variable under control reaches either the high or the low limit previously established for it.

2.43 By pressing a suitable button, the operator can acknowledge his awareness of the presence of the plant error. The button will silence the audible alarm and cause the light to cease flashing. However, the light will continue to shine steadily as long as the variable is out of limits.

2.44 Once the variable is back within limits, the light will go out.

2.45 Should still another variable within the same unit go out of limits while the light is burning steady. it will begin flashing again and the audible alarm will sound again thus indicating a new alarm condition to the operator.

2.46 The location number and designation of the out-of-limits variable (i.e., high or low) may be typed out on an optional alarm printer or displayed on

the CRT screen whenever the variable goes out of limits.

2.47 The optional alarm printer, if used, should be in addition to the logging typewriter described above.

2.48 In order to bring a new operator up-to-date following a shift change, a method should be provided for reviewing the situation existing at that time as regards alarmed variables. One possibility is to have the alarm printer print out all off-limit variables on demand. Another possibility is a button to reset all alarmed conditions and thus allow the alarm unit to reactivate any alarm conditions still present at the monent this button is pressed.

2.49 A separate audible signal and flashing light system shall be provided as part of the alarm unit to notify the operator of computer system errors or failures such as included under Item 2.9.

2.5 The Computer

The computer in its various parts must perform modification operations on input data, preparation of data for subsequent logging if used, and the control computations themselves.

In the interests of reliability and economy, it should be kept as simple as possible in design and construction.

2.6 Control Outputs

System outputs from the computer as control corrections will be used to adjust the settings of the primary control values of the process in order to bring the various process variables into line with their desired values as previously set by an operator or by a supervisory control computer. This may be accomplished in several ways. Some of these are as follows:

2.61 A d.c. signal in the range of 1-5 ma corresponding to standard electronic analog control loops. Such signals would be fed to conventional current to pressure transducers and then to standard pneumatic valve actuators.

2.62 A discrete or digital pulse type signal, the number of pulses to represent the magnitude of the signal involved. Such a system would use a constant pulse rate and could determine pulse number by counting down a binary counter containing the control correction value.

2.63 A single discrete or digital pulse type signal whose pulse length or pulse height would be representative of the magnitude of the correction imposed.

2.64 Contact closures to operate alarm systems and other on-off devices.

2.7 Manual Backup

As mentioned in Item 2.24, it is imperative that a method for manual operation of the plant be provided to allow for continued operation of the plant during as many as possible of the potential computer system interruptions. It should have the follow features:

2.71 Automatic switching from direct computer control to a "locked" condition whenever computer or input signal failure of any kind occurs. This "locked" condition shall preserve the last set value of control output so as to maintain the last valve position until released by a subsequent manual correction or by the computer returning to control of the process. In some few cases, the valves may be required to go to some preset "safe" position upon computer failure.

2.72 A backup power supply independent from the rest of the computer so that computer power failure cannot interfere with computer or manual control backup operation.

2.73 In case analog outputs are being used, the backup valve control device could be a potentiometer whose position and hence output voltage could correspond to total valve air signal. It would thus correspond roughly to valve position.

2.74 For digital or pulse output signals, an "inching"

button or switch, which would apply a train of pulsesto the valve motor during the time the switch or button were engaged, could be applied. A single button could be multiplexed by means of a selector switch.

2.75 Some method of rough indication (~5%) of valve position would be extremely helpful in startup operation of the unit and in putting the computer "on line" for control. Such a device might operate from a differential transformer on the valve stem. Any other suitable method of indication could also be used.

2.76 Some method of assuring "bumpless" transfer from manual to computer control such as a fast reset while in manual position.

2.8 Systems Tests

In order to assure the operator of the continued satisfactory operation of system components a series of tests of the such operation will be undertaken periodically as part of the regular routine of the computer based industrial control system. These shall include:

2.81 Watchdog timer. Each processor of the system shall have associated with it a separate mechanical or electrical device which will be reset periodically by the computer system. Should the computer fail to reset the device at the conclusion of the stated

period it shall independently initiate the switch-over or fail-safe operation prescribed for computer mainframe failure in the system. The period of time involved will normally be of the order of one or a few seconds.

2.82 Trial Computations. Every computing device in the system will have incorporated as its lowest priority function a set of trial computation procedures capable of testing each of the computational elements and storage devices of that computer. These will be run continuously during all control systems "idle" time. Failure to obtain a "correct" answer at the conclusion of any one of these tests will initiate a repeat of this test. Failure to succeed after three of these repeats will initiate the prescribed switchover or fail-safe operational procedure.

2.83 Closed Loop Test. Each separate multiplexer unit shall have associated with it, a "closed loop" system which will accept a signal from a field mounted device which is periodically reset by the computer according to a preset pattern which is initiated in turn by the analog input from the field. Thus a self-perpetuating loop is obtained. This last shall be associated with an error sensing device and a timer in such a way that a failure

to obtain the correct value from the field to an accuracy of 0.25% or to obtain such a value within a specified period of time (one or a few seconds) will initiate an alarm at the operators alarm unit (Item 2.4).

3. DESIRED MACHINE CHARACTERISTICS

3.1 Machine Availability for Continuous Operation

As mentioned earlier, machine availability or time actually "on-line" is probably a more important consideration than reliability per se. On-line availability of present optimizing or supervisory control computers is in the range of 99.5% of plant on-stream time. Such performance still means approximately 40 hours per year downtime. This would be entirely unacceptable where plant control is being entrusted completely to the computer.

An availability of 99.95% of possible plant on-stream time or better (with a resulting downtime of 4 hours or less per year) would on the other hand probably be allowable. This would be especially so if the auxiliary devices (separate logger, standby manual operation of facilities, etc.) previously mentioned are available in the computer system to allow satisfactory standby operation during repair.

As also mentioned earlier, computer downtime must include the time required to diagnose a machine failure and affect repairs to the computer. Computer maintenance checks which cannot be coordinated with regular plant off-normal operational procedures must be included in the four hour yearly allowance. Thus, ease of maintenance combined with a well planned set of diagnostic instructions and procedures for the maintenance man is vitally important. In order to achieve the above, a desirable goal would be not more than

one machine failure per year (8000 hours).

3.2 Read-out for Trend Recording to Accompany the Logging Option

Operation of a chemical or petroleum plant is a very involved procedure during which the operator requires a knowledge of the plant's present status and its immediate past history. Analog "trend recording" of selected plant variables can therefore be an important and valuable tool. For this reason, we wish to consider the possibility of providing similar recordings with the control system under discussion here.

3.3 External Priority Interrupts

When used in conjunction with a large optimizing or supervisory control computer, the direct digital control computer should have the ability of activating the priority interrupt feature of the large machine. The supervisory control machine could thus take corrective action for an off-normal situation detected by the direct control computer.

3.4 Automatic Nondestruct Shutdown of Memory

It is imperative that means be provided to insure against the possibility of destruction of the program or data memory locations during machine shutdown or temporary failure, or in the event of external power failure.

3.5 Memory Protect

Provision shall be made so that certain areas of memory (both fast memory and rotating bulk memory) can be designated by hardware operations as "protected" in such a way that one cannot read into these areas from other unprotected areas. Protection may be by single words or by blocks. The system should be able to protect up to one-half of available memory.

3.6 Parity

It should be possible to perform parity tests on all internal and external transfers of data throughout the computer and the data system. In the event of a detected parity error, the operator should be notified by the alarm functions. Shutdown or transfer should not be initiated except by operator request.

3.7 Floating Point Hardware

The wired-in capability of carrying out floating-point arithmetic and related operations should be provided for all Level 2 and higher degital computers.

4. OTHER TECHNICAL CONSIDERATIONS AND DEFINITIONS

4.1 System Grounds

The computer vendor must detail his minimum requirements for a system ground or grounds for the computer system for compatibility with the rest of the instrumentation system.

4.2 Terminal Blocks

4.21 The instrumentation systems to be associated with each direct control computer will normally be installed in such a way that signals from all operating plant units will be made available to the computer at a "terminal block" in each control room or remote multiplexer location. The vendor will then only be responsible for transferring the signals from the terminal block to and through his supplied equipment. Normally this control room "terminal block" will be the input terminal strip on the first unit of the computer or an associated multiplexer.

4.22 Similarly, the vendor will normally be responsible for transferring the computer output signals only as far as the output terminal block of the respective control room or remote multiplexer installation for control signals to the plant operating units. Again this output terminal block will normally be the output terminal strip of the last computer unit.

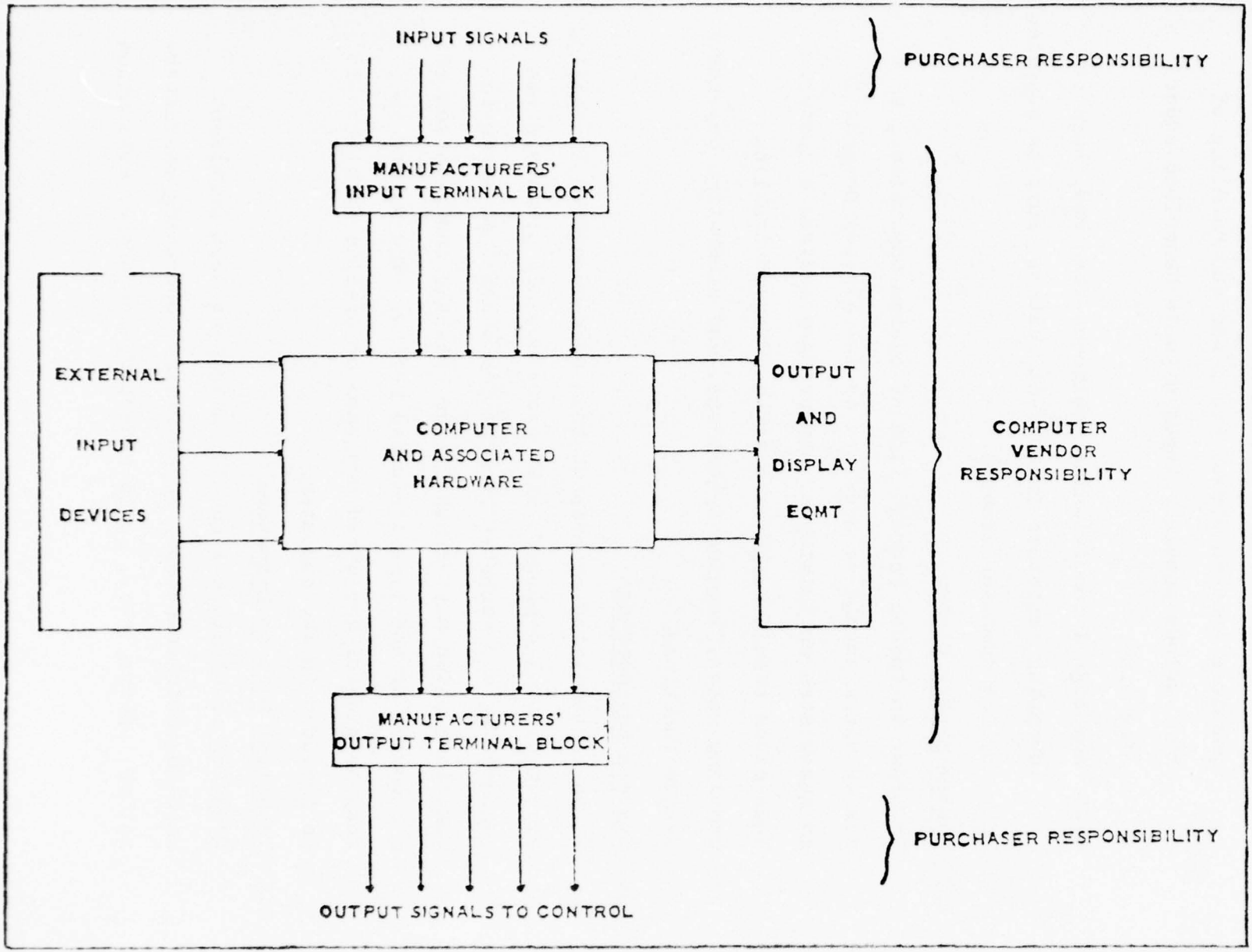


FIGURE 7

COMPUTER-INSTRUMENTATION
COORDINATION RESPONSIBILITY

4.23 Figure 7 should further clarify Items 4.21 and 4.22.

4.24 Vendor should present his recommendations concerning possible lead-in wires, etc., for the handling of the various possible input signals described under Item 2.11.

4.25 Any signal conditioning equipment required, such as dropping resistors in current systems, must be supplied by the computer vendors.

4.3 System Master Clock

In order to insure coordination of system operations, a master clock should be provided if not already present in an associated optimizing or supervisory control computer. This clock should supply the signals for controlling sampling rates, logging operations, and other time dependent system functions.

4.4 Systems Engineering

It is contemplated at present that user companies installing direct control computers will carry out all plant systems engineering and computer programming other than wired-in functions which may be associated with the installation of a computer of the type discussed herein. The vendor is responsible for any wired programs or special functions to be included in the computer.

4.5 Technical Data on Computers

Computer manufacturers should include as much pertinent data on their computer's capabilities, operating characteristics, speeds, etc., as is necessary for proper evaluation

of the proposed machines. These data must include at least the following:

4.51 A full discussion of machine organization and method of operation.

4.52 Speed of operation of all associated input and output equipment.

4.53 Physical sizes, weights, power supply needs, required floor space, etc.

4.54 Drift, temperature sensitivity, frequency response, and common mode rejection capabilities of system input amplifiers.

4.55 Complete reliability analysis.

4.6 Thermocouple Reference Junction

If thermocouple reference junction units are to be provided by the computer manufacturer for the thermocouple inputs, they shall have a temperature stability of at least $\pm$ 0.2°F. Vendors should specify the number of thermocouples which can be included in each reference unit.

4.7 Environment and Electrical Classification

4.71 The computers will normally be installed in air-conditioned control rooms of standard, petrochemical plant type design. The atmosphere in these control rooms will be relatively good, but cannot be maintained completely free of dust and may, at times, be subject to some small amounts of hydrocarbon, sulfurous, ammoniacal or halide vapors (all less than 5 ppm).

4.72 The control rooms will commonly be pressurized and be classified as "general purpose" electrical areas. However, individual requirements vary and should be investigated. A Division 2 classification is generally the most severe required.

4.73 Vendor should specify any environmental limits required for the machine such as maximum and minimum allowable ambient temperatures, humidity ranges, etc. A desirable temperature range would be to be operable from 50-110°F. with no damage to 130°F.

4.74 In some cases, plant control rooms may contain small amounts of hygroscopic dusts. For such specific situations, tropicalization of the electronic equipment would be very desirable. In addition, an open type configuration of the equipment such as required for easy maintenance (Item 1.213) would be quite helpful for cleaning of this equipment of deposits which may form from hygroscopic dusts.

4.75 Remote multiplexers will not normally be mounted in air-conditioned and pressurized areas. Therefore these must be protected by enclosures or other devices capable of providing necessary protection from plant moisture, noxious vapors, and temperature variations.

4.76 All field data and control systems should operate at "intrinsically safe" levels of electrical voltage and current or power as prescribed by appropriate standards.

5. PROCESS INDUSTRY NEEDS FOR DIRECT CONTROL COMPUTERS

5.1 System Size

It is anticipated that there will be process industry requirements for computers of the type discussed herein in sizes equivalent to the following number of control loops.

5.11	16 loops
5.12	32 loops
5.13	64 loops
5.14	128 loops
5.15	256 loops

Greatest demand should be for the three medium sized machines with somewhat less interest in the very large or very small sizes. Expandability in the field would be a very desirable additional feature.

6. RELATED SYSTEM CONSIDERATIONS

6.1 Noise Rejection

Remote multiplexing of plant signals should reduce greatly the problems of overall system noise rejection since high level digital signal will normally be carried in long signal leads. However, users must continue usual good wiring practices in obtaining analog signals from the field and carrying them to the nearest field mounted multiplexer. Use of twisted, shielded leads, of proper line balancing, and good grounding practice must be investigated and used where appropriate.

In addition it is necessary that noise rejection capabilities of the multiplexer data-input section be exceptionally high. Therefore, it is requested that the input data section have at least the following characterisitics:

6.11 Ability to reject up to 10 volts of common mode noise at 60 cycles AC when imposed across a 5 mv DC signal with a resulting shift of 1μ volt/volt or less. There should be no damage when voltages as high as 50 volts are applied.

6.2 Sampling Techniques

In the interest of the ultimate simplification of programming systems, all sampling of process variables should be sequential on each remote multiplexer in turn on

initiation by a single signal from the system timing device. No random sampling capability should be provided. Overall sampling speeds and analog to digital conversion speeds must be fast enough so that all variables are sampled at the speed needed for the fastest response function of the all system.

6.3 Direct Memory Access

To prevent the required fast sampling speed of Item 6.2 from overloading the computational facilities of the system, all computers should have direct-memory-access devices to permit reading of input values directly into memory rather than through the arithmetic unit.

II. SOFTWARE CONSIDERATIONS

7. HANDLING OF PROCESS I/O BY THE COMPUTATIONAL UNIT

7.1 Process Measurement Inputs

7.11 Scheduled Scan: It should be possible to specify an internal computer scan frequency or processing interval for each input. Completely random specification of frequency will not be necessary if enough different time classes are included.

It should be possible to distribute the scanning of the inputs in a given time class over the period of the time class so that the computational facilities are not overloaded at certain times.

For example, if phasing were not done on a system containing 0.25, 1, 5, 15, 30, and 60-second time classes, they would all come due every minute, possibly causing the scanner to fall behind. Phasing however, must not interfere with the above grouping requirement.

7.12 Validation: The input value of each scanned input should be checked for validity limits and/or illegal configuration.

It should be possible to remove an input from interval computer scan, or limit checking when it has failed validity limits for n consecutive times,

where n is a system generation parameter.

It should be possible to output a message with the out-of-limit value when a point has failed validity limits n consective times.

7.13 Conversion:

7.131 Compensation: input readings should be compensated for zero offset before applying conversion equations.

7.132 Standard equations: Provision should be made for several different conversion equations. for example

(1) Thermocouples including the following:

Iron-Constantan

Copper-Constantan

Chromel-Constantan

Chromel-Alumel

Platium - Platinum Rhodium (10%)

Platinum - Platinum Rhodium (13%)

Conversion co-efficients should be supplied which will provide errors within the limits of ISA Standard C96.1. Input readings should be compensated for thermocouple reference before conversion.

(2) Linear conversion of the form MX + B

(3) Polynomial conversion

(4) Flow conversion incorporating a square root function

a. Liquid Flow:

Uncompensated

Temperature Compensated

Gravity Compensated

Temperature and Gravity Compensated

b. Gas and Steam Flow

Uncompensated

Pressure Compensated

Temperature Compensated

Gravity Compensated

Combination of above

(5) Special equations: user should have the ability to specify any other conversion, such as table look-up for code conversions, and to relate scan frequency to filter and/or conversion processing interval up to a certain predetermined number.

7.14 Filtering: Each input should have the option of having its value filtered using a digital filter factor specified on a point-by-point basis. The user should have the ability to specify any filter equations, and any filter processing interval.

7.15 Alarming:

a. Each input should have the option of having

its value checked against both high and low limits and a rate of change limit. When limit violations occur it should be possible to:

(1) print a message

(2) cause the execution of a program

(3) open or close a contact (user should have the ability to specify the list of contacts)

(4) any combination of the above

7.2 Process Outputs

7.21 Of the two common internal forms of outputs (absolute and incremental) the incremental value is preferred.

7.22 Output Generation: The generation of outputs should allow for demand outputs or the specification of frequency and phasing of each output.

7.23 Deviation Limits: Deviations are defined as the difference between the current output value and the newly calculated output value. Maximum absolute deviations may optionally be specified for each output which are the limiting values of any change in an output.

If the deviation limit is exceeded, a program linkage to a specified program can be established. Dead-band capability should be provided.

7.3 Event-Oriented Input

Interrupts should be caused by changes in state of single bit inputs from one logical condition to the other, but not the reverse.

8. MAN/COMPUTER COMMUNICATIONS RELATED TO OPERATION AND CONTROL OF THE PLANT

8.1 General

A response should be given to the requester to indicate either a valid or invalid request was made. If an invalid request has been made, the system should indicate the reason for trouble.

8.2 Process Operator and/or Supervision

8.21 Active entry functions:

These operations should be available to change the operating status of the system. The implementation of the operator entry functions should be done in such a way as to require his performing certain additional coded (keyed) actions so as to minimize the possibility of inadvertent entry of erroneous information. All parameters available to the operator should have a range of acceptable values and the operator should be notified when his entry is not within the acceptable range.

8.211 Alarm Scan: The operator should be allowed to:

(1) Activate or deactivate any alarm.

(2) Change the high and low alarm limits.

(3) Change the rate-of-change limits.

8.212 Logging: The operator should be allowed to:

(1) Add/delete a variable to/from a log.

(2) Request logs on a timed frequency or on demand. This includes designation of an interval.

8.213 Trend recording: The operator should be allowed to specify the pen number and variable along with the desired frequency, the desired range and intercept.

8.214 Control: Those parameters which require knowledge of control theory and timing should not be considered as operator functions. He should be allowed to:

(1) Change the set point of a loop.

(2) Activate or deactive a loop.

8.215 Demand Functions: Certain functions, including user application programs, should be made available to the operator on demand. He should have the capability to:

(1) Select the functions

(2) Specify a specific time for execution or

(3) Specify a time interval for execution or,

(4) Demand immediate execution.

(5) Terminate the function.

8.216 Miscellaneous: The operator should be able to:

(1) Set the time and date.

(2) Activate or deactivate selected peripheral I/O devices.

(3) Enter pre-specified process data.

8.22 Passive Entry Functions:

These operations should be made available to interrogate the operating status of the system.

8.221 Process I/O scan: The operator should be able to list or display:

(1) The status of a variable (active/inactive)

(2) The value of a variable or of a raw data input.

(3) The identification of all variables deactivated.

8.222 Alarm scan: The operator should be permitted to:

(1) List or display the alarm status of any alarmed variable.

(2) List the identification along with limits and value of those variables currently in alarm.

8.223 Logging: The operator should be allowed to:

(1) Obtain a list of those logs currently inactive.

(2) Obtain a list of those logs currently on automatic request along with the interval for each.

8.224 Trend recording: The operator should be able to obtain the current status of each trend pen including the identification and present value of the variable along with its range, its intercept and its scan frequency.

8.225 Control: The operator should be able to list or display:

(1) The current set point of a loop,

(2) Whether a loop is active or inactive.

(3) The control limits.

8.226 Demand functions: He should be allowed to list a menu of the demand functions, their parameters with regard to demanding a program, and status (active/inactive).

8.227 Miscellaneous--he should be able to demand the current:

(1) Time and date.

(2) Status of selected I/O peripherals.

(3) Value of preselected process data.

BEST AVAILABLE COPY

8.3 Control Hardware

8.31 Active Entry Functions

8.311 Process output: The control engineer should be able to activate or deactivate and control any digital/analog device. However, these actions should not be permitted unless the control loop corresponding to the device is deactivated.

8.312 Control: The engineer should be able to change all parameters having to do with a control loop, such as algorithm constants, type of algorithm to be used, maximum allowable change, control frequency, cascading optings, etc.

8.313 Process I/O scan: The engineer should be able to change other variable parameters, such as computational frequency, conversion coefficients, validation limits, conversion type, filter constants, engineering units, flow coefficients, type compensation, and others as necessary.

8.314 Data age checking: He should be allowed to:

(1) Specify any variable for age checking as a validity check.

(2) Specify the age limit for any variable being checked.

8.32 Passive Entry Functions

8.321 Digital I/O: The engineer should be able to

BEST AVAILABLE COPY

obtain a print-out or display of the configuration of a specified group or bit (s).

8.322 Control: He should be able to obtain a list of all parameters and their values, having to do with a specified control loop.

8.323 Process I/O scan: The engineer should be able to list and/or display the value of any variable parameter, such as conversion coefficents, validation limits, conversion type, filter constants engineering units, flow coefficients, type of compensation, and others.

4. Programmer

8.4 Active Entry Functions

8.411 The programmer should be permitted to change the contents of from 1 to n consecutive locations in core or bulk storage through the programmer's I/O device, in decimal and in the number system used by the system assembler. Any changes to either core or bulk storage should be accompanied by a print-out of the contents of the altered location(s) before and after the change.

8.412 The programmer should be able to cause the transfer of from 1 to n locations to/from core

or bulk storage.

8.413 He should be able to read (card or tape) and store 1 to n locations into core or bulk storage.

8.42 Passive Entry Functions

8.421 The programmer should be able to list the contents of from 1 to n consecutive core or bulk storage locations on the programmer's I/O device in decimal and in the number system used by the system assembler.

8.422 He should be able to punch (on cards or tape) from 1 to n consecutive locations of core or bulk storage.

8.43 Program Loading and Linking: The programmer should be able to load (via cards or tape) and link programs to the operating system on-line, and enter the parameters necessary to make the program function in the system.

9. OPERATING SYSTEMS

9.1 Interrupt Handling

9.11 Definitions: For the purposes of this specification, "interrupt" shall refer to suspension of the execution of a routine as a result of a hardware or program generated signal. Interrupt handling is the procedure which the operating system follows when responding to an interrupt.

9.12 Interrupt levels: In cases where the hardware configuration has a number of interrupts, the operating system should be capable of responding to these according to the assigned level or priority of each interrupt.

9.13 Interrupt conditions: With respect to the current routine and an interrupt, four conditions can exist:

9.131 Condition 1: No interrupts are present.
Action: Continue the current routine.

9.132 Condition 2: Current routine is of a higher priority than the interrupt.
Action: Continue the current routine.

9.133 Condition 3: Current routine is of same priority as interrupt.
Action: Continue the current routine.

9.134 Condition 4: Current routine is of a lower priority than the interrupt.

Action: Suspend the current routine and take whatever measures are necessary for subsequent continuation, then process the interrupt.

9.14 Interrupt Processing: In processing an interrupt, the system should at least accomplish the following:

9.141 Since it is impossible for an executing routine to anticipate the instant when an interrupt will occur, the operating system should suspend the execution and save whatever data is required for the suspended routine to continue at a later time.

9.142. Take necessary steps to delay processing of any interrupt states of a priority equal to or lower than the currently active state.

9.143 Give control to the interrupt response program for the interrupt currently being recognized.

9.2 Program Scheduling

9.21 Levels: The operating system design should be such that the user can assign priority levels for the execution of application programs.

9.22 Enabling: The application programs shall be able to enable/disable the interrupt system, at least to the extent of controlling interrupt connection

to other application programs.

9.23 Realtime Clock, Power Fail-Restore: The system interrupt structure should give highest priority to the power failure-restore interrupt, if this option is provided. Next highest priority should be assigned to the system realtime clock. All other peripheral and process interrupts should be of lower priorities.

9,24 Execution Priority: All programs are assigned an execution priority. When more than one program is scheduled, the comparison of these priorities is the determining factor for one program taking precedence over another. The operating system priority structure should be designed such that interrupts and programs are merged into the overall priority scheme to insure that use of computer time is handled on a priority basis.

9.25 Execution Scheduling: It should be possible to schedule the execution of a program in one of the following ways:

9.251 Interrupt active: when recognized, schedules the execution of a program.

9.252 Passage of time: it should be possible to schedule the execution of a program on a timed frequency or at a time of day. The requestor

BEST AVAILABLE COPY

should be able to specify:

(1) time of day for execution, or

(2) time increment for execution with a starting time, and

(3) execution priority

9.253 Program request: An executing program should be able to schedule the execution of another program. It should be possible for the requesting program to specify the execution priority of the requested program. With this capability, the requesting program should have the following options available for scheduling the execution of requested programs:

(1) immediate execution (higher priority)-- with optional return to requesting program after completion.

(2) deferred execution (lower priority) -- with no return to requesting program. Execution of requested program follows completion of the requesting program.

9.254 Program Suspension: Discontinue program execution for a specified time interval, without disturbing the logical continuity of the program.

9.3 Memory Utilization

9.31 Allocation: The operating system should have the capability of dynamically allocating and assigning available core for temporary use to requestors. This allocation should be done according to a requesting priority. When requesting core, the requestor should be able to specify:

a. amount of memory needed

b. priority of request for memory

The operating system should then satisfy requests for memory allocation according to the requesting priority.

Allocatable memory may be partitioned according to program priorities and high priority memory would not be available to lower priority requestors. The actual dimensions of partitions would be installation parameters and should be done in such a way as to make memory available to satisfy the needs of higher priority requestors first, and thus maintain the overall integrity of the priority scheme.

It should be understood that a requestor returns allocatable memory, since the operating system cannot sense when the requestor is finished with it.

9.32 Roll-out: Very often the need develops to execute a high priority program while allocatable memory is

occupied with lower priority programs. When this occurs, the operating system should be capable of suspending and saving the lower priority programs, freeing allocatable memory for higher priority use.

9.4 Input/Output Handling

9.41 General: The operating system should include the necessary drivers to operate all I/O equipment (process and non-process) which might be included in the system.

9.42 I/O Requests: Should have the following characteristics

9.421 Consistent format: The requirements for making I/O requests should be organized in a consistent format. The syntactial conventions, which must be followed by requestors should be consistent throughout for all I/O requests.

9.422 The requestor can specify an I/O device in a symbolic way. It should not be necessary for the requestor to know a device number and other hardware characteristics peculiar to a device, such as channel, line, and equipment numbers.

9.423 Priority: Requestor specifies the priority at which the request is to be serviced. This is done at execution time.

9.424 I/O List: Requestor specifies symbolically the location of the I/O list.

9.425 Completion: Requestor should have the option of

suspending execution of the calling program until the completion of I/O.

9.426 Buffered/non-buffered: Where available, the requestor should be able to specify that I/O be accomplished either buffered or non-buffered.

9.43 Processing I/O Requests: The operating system should be the interface between I/O requests and the device drivers. Checking the validity of I/O requests and insuring that they are satisfied in order of priority are functions of the operating system. Conversely, the requestor should not be required to take any action which might involve status checking or I/O timing considerations.

9.5 Background Processing (Level 3 and higher systems)

9.51 Definition: A background program is defined in the glossary as a program of no particular urgency with regard to time and which may be pre-empted by a program of greater urgency and priority.

9.52 Job Processor: If the operating system includes background capability, the job processor should be called or scheduled on a demand (interrupt) basis by a user.

The job processor should have control of all program development, debugging, and other utility programs available to the user.

In addition to having control of vendor-provided utility programs, the job processor should be capable of creating and maintaining a library of background programs generated by the user. The background job processor should, by the nature of the function it performs, operate at the lowest priority.

9.6 On-Line Diagnostics

9.61 Definition: Most computer programs employ a measure of dignostics to test for system malfunctions. For this specification, diagnostics considered are those which the operating system performs in order to maintain a properly functioning system.

9.62 Request: The operating system should verify, in so far as practical, the parameters required by requests to the system. When the system detects an error condition, it should take the necessary action to protect the system and communicate the error condition back to the requestor.

9.63 Timing: All events should have a timing constraint placed upon them. This would be a parameter defined at load time. The operating system should take action to terminate the event when the time limit is exceeded. This commonly occurs when an I/O device fails and leaves a program suspended, waiting for completion of I/O or when a program

hangs in a loop.

9.64 I/O: The operating system should provide for suitable alternative action to be taken whenever a primary device fails. Ordinarily, this action would not involve notifying the requestor. However, if the situation is irrecoverable, the requestor should receive some indication of the condition in order that he might take some additional action.

9.65 Recovery: The operating system should, on regular intervals, make some diagnostic checks, such as checking system constants, to determine that the operating system is intact. It should have automatic recovery capability to be used whenever unusual or abnormal conditions are sensed within the system.

9.7 Expandability

9.71 Modularity: The system should be designed and organized with the expectation that it will be expanded. The user should have options available to organize a system to meet his particular requirements while minimizing the inclusion of functions which he does not require.

9.72 Field Expansion: Field expandable hardware components should be supported by comparable operating system expandability.

10. SYSTEM GENERATION

10.1 Introduction

System Generation (definition):

"The task of establishing an operational system composed of:

a. installed hardware,

b. vendor-supplied programs, and

c. those programs written especially for the application, which will be expected to function in either foreground or background mode on the application computer, while the process is on-line."

Certain utility, diagnostic, or various other categories of programs and any special hardware required by them may also be operational in the background mode or when the application computer is off-line.

Program Development is the series of steps between defining the problem and having a functioning program. Testing and debugging begin when the program has been prepared in source form.

Some programming errors are detected when the program is compiled; others require execution of the program, possibly with simulated inputs and outputs, traces and dumps, before yielding to detection.

When the program functions as intended, it is then installed into the system. There will usually be changing system requirements which preclude permanent installation of a program. This sequence may be repeated several times, as part of the debugging process.

This section will not deal with the way the compiler for the high level language(s) transforms the source statements into executable object code. However, the treatment of all source statement types by the compiler as related to the administration of the object code, such as converting (data), loading, linking and relocating, are dealt with as deemed necessary.

10.2 Software Development Considerations:

10.21 User Application Programs

10.211 Program Generation

(1) Coding or statement preparation;

(2) Source input preparation (i.e., key punching; etc.);

(3) Compile/assemble

10.212 Data Preparation

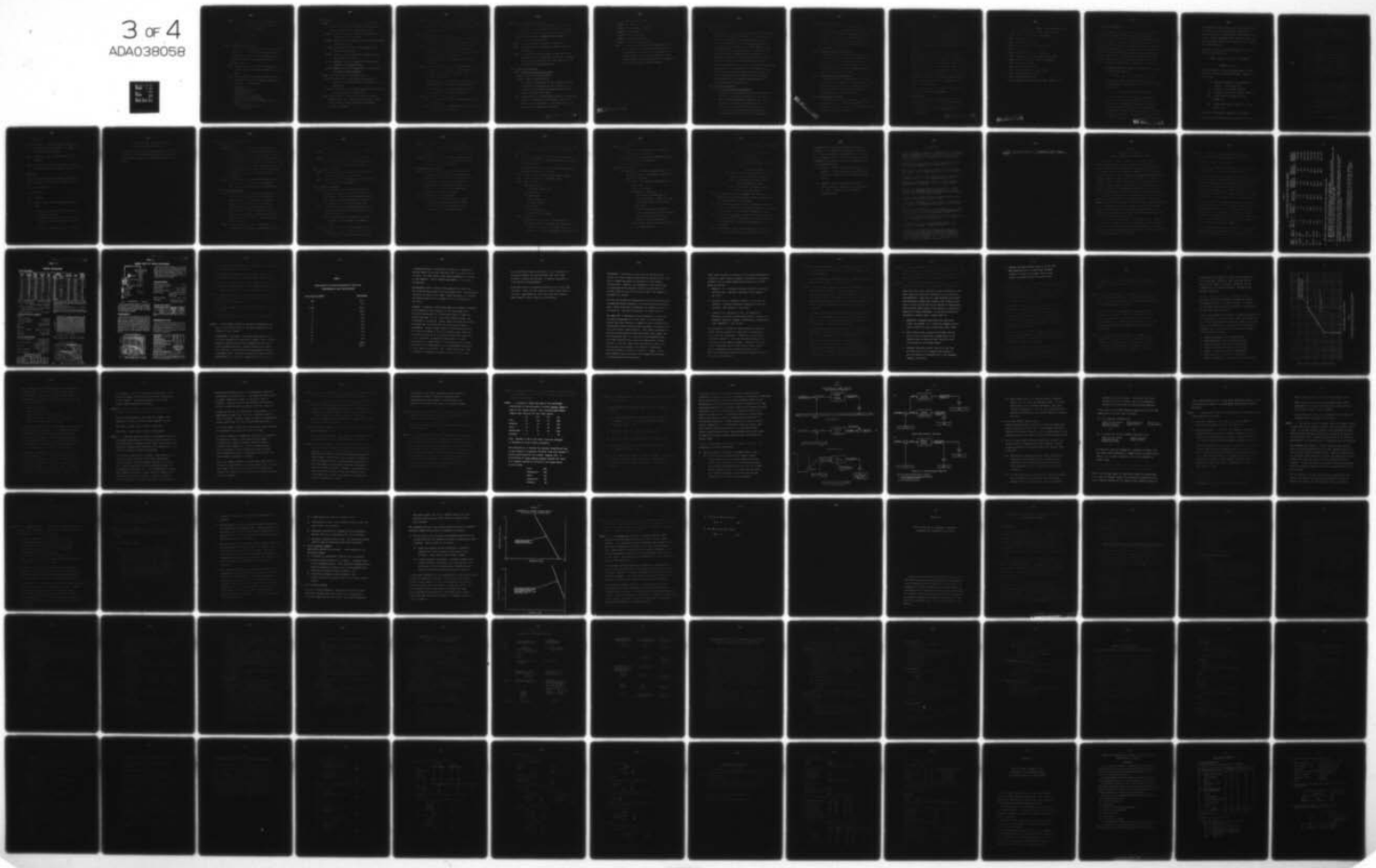
(1) For On-Line use

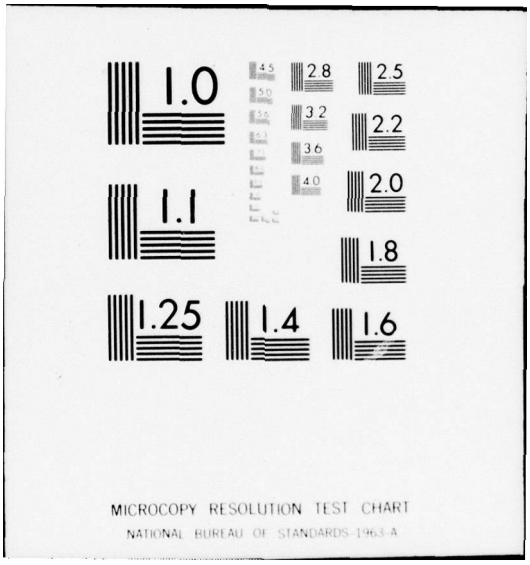
(2) For testing use

AD-A038 058 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIO--ETC(U)
JAN 77 N00014-76-C-0732

UNCLASSIFIED NL

3 OF 4
ADA038058




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

10.213 Testing: Both partial and complete systems

(1) Loading (and linking as required);

(2) Simulation of process;

(3) Tracing:

(a) Arithmetic

(b) Logical

(4) Debugging

10.22 Vendor-Supplied Programs

Customizing vendor software to needs of a specific application. The program-development programs: compilers, assemblers, loaders, emulators, etc. The on-line programs, operating systems, scan packages, etc.

10.221 Remove modules which are not applicable, device drivers not used, conversions not used, etc.

10.222 Supply special information about hardware configuration (see also D, Linking Software to Hardware). Examples are:

core size;
disk capacity;
tape configuration;
input options;
output options;
interrupt trap addresses;
special indexing addresses;
other special use locations;
data acquisition options (direct memory access, etc.);
signal conditioners
(1) logical inverters
(2) shifters

10.3 Loading

10.31 There should be the ability to generate object coding with appropriate "header" information so that subsequently the object code can be:

10.311 Loaded in core as compiled with minimal instruction from its header;

10.312 Loaded in next available core (considering pages, odd, even, etc.);

10.313 Loaded in core after a halt pending an on-line origin directive;

10.314 Loaded in object-image onto secondary memory as directed in header;

10.315 Loaded as in 10.314 except in "next available segment" of secondary memory;

10.316 Loaded as in 10.314 except after a halt pending an on-line origin directive.

10.32 There should be the ability to copy:

10.321 Relocatable program images from secondary memory into reloadable form on compatible input media;

10.322 Data images from secondary memory into reloadable form on compatible input media.

10.33 There should be provisions to reload previously dumped relocatable program images and data images into secondary memory beginning at specified starting locations.

10.34 There should be provisions to load compiled relocatable program images and compiled or assembled data images onto secondary memory at a suitable starting location.

10.4 Linking Software to Hardware

Ability to inform the system through a mechanism, such as a symbol table available to the compiler, of:

10.41 The physical termination identification of each peripheral industrial application component (process I/O device) with which it must communicate: (sample specification statement)

Program Identifier = Physical Termination Identifier (ID)

10.42 The physical identity of the communication channel(s) over which information will be exchanged with each industrial applications component: (Sample specification statements)

Program Identifier = (industrial application component) via (communication channel, identifier)

Data to/from (industrial application component) via (communication channel, identifier)

10.43 The physical identification of:

10.431 each separable interrupt level and its program identifier;

10.432 each separable interrupt point at a given level and its program identifier.

10.5 Language-to-Language Linking

In order to provide maximum flexibility to the programmer, he should be allowed to choose the language best suited to stating the task to be performed. He should be permitted to intermix these language statements within one program.

10.51 Industrial Process Language to Assembler;

10.52 Industrial Process Language to Problem Oriented Languages;

10.53 The Industrial Process Language Complier should tolerate "Foreign" statements in the above languages and permit the Assembler and other Compiler to analyze these for correctness, etc.

10.6 Areas of Improvement

10.61 Greater modularity of software

10.611 Operating System Software

10.612 Other Packaged Software

The vendor documentation of these packages should be adequate for determining how to remove task modules and associated subroutines which the user may not require in his system.

10.62 There should be a two-way-entry method of determining whether a subroutine can be dropped from the library, i.e., by subroutine, what tasks use it, and by task, what subroutines are used.

BEST AVAILABLE COPY

Examples of modules are:

10.621 I/O drivers

10.622 Data conversion routines

10.623 Scan programs

10.624 Format conversions

10.625 Arithmetic subroutines

10.626 Parameter-passing options

10.63 A system generation scheme is required for small object systems, such that the object system can be generated on some other, larger system. The larger system may be located locally, or remote from the object machine, and may be of a different make or model from the object machine.

BEST AVAILABLE COPY

11. SYSTEM MAINTENANCE

11.1 Issue 1

Reliability and maintainability are interacting characteristics in a system and result in a single feature: system availability. The implementation of system maintenance depends on the availability required of the system, and the extent of the task depends on the reliability of the component parts. In the sections that follow, the implementation of maintenance is ommitted on the basis that it is peculiar to a system and must be developed for that particualar system.

The objective will be to organize a preventive maintenance program where all maintenance is on a scheduled basis. Nevertheless, provision should be made for diagnostic aids to allow rapid replacement of failed equipment. The remainder of this section assumes the use of planned preventive maintenance.

11.2 Hardware Maintenance

11.21 C.P.U. and Standard Peripherals

Diagnostic testing programs shall be available for the C/P.U. and standard peripherals. As this is the basic vendor-supplied equipment, this can be regarded as a mandatory requirement. It is desirable that two forms of test be available:

a. Simple on-line fault indicating programs.

b. Comprehensive off-line fault finding programs.

11.22 Process Equipment

Diagnostic programs should be provided for the process equipment under computer control. Basically, this may be an equipment logging and monitoring system. As a minimum, the faulty control loop should be indicated, and in some circumstances it may be desirable to pinpoint the faulty equipment, i.e., valve, sensor, etc.

11.23 Economic Preventive Maintenance

To ensure that the preventive maintenance program is carried out economically, it is suggested that the logging and monitoring system be used to develop operational reliability data on the installation and that this data should be used to modify the maintenance shcedule.

11.24 Computer Assisted Maintenance

It may be desirable to implement, on a computer, programs that will supply or process data to assist maintenance personnel in their tasks, e.g., list likely equipment faults, suggest the type of maintenance--repair or replace, etc.

11.25 Spare Parts Inventory

This could usefully be performed by a computer. Stock levels to be held could be produced by the computer, based on quantity in use, past reliability history, and order lead time.

BEST AVAILABLE COPY

11.3 Software Maintenance

Adequate provisions must be made to protect software integrity. Controls should be built into the system to restrict the ability to destroy data or programs. Similarly, the incorporation of program revisions or updates must be organized to reduce the possibility of erroneous or incompatible software existing in the system.

The implementation of software to handle the man-machine interface should be such as to minimize the possibility of the operator carrying out unauthorized operations (while this is primarily needed for product and personnel safety reasons, improper operation of equipment may reduce reliability). If it is possible in a system to reroute a device, (e.g., a computer output fails and there is a spare output available), then it should be possible to easily amend the reference to that device.

On-line routines shall be available to allow initialization of replacement equipment.

11.4 Documentation

Basic to any maintenance function is the system documentation. Note also that this documentation must be maintained to reflect the current status of the hardware and software throughout the the life of the system.

BEST AVAILABLE COPY

The following list constitutes a minimum documentation package that should be provided with a system:

11.41 System Description and Objectives

11.42 Reference Manuals

11.43 Design Specifications

11.44 Acceptance-Testing Documentation

11.45 Warranties

11.46 Hardware Maintenance Handbooks

11.47 Contract Maintenance Schedules (if used)

11.48 Spare Parts Lists, Prices and Sources

11.49 Programming Manual

11.50 Program Descriptions

11.51 Program Listing and Flow Charts

11.52 Initializing Procedures

11.53 Operating Procedures

11.54 Program Sources (Users' groups, authors, etc.)

BEST AVAILABLE COPY

12. CONTROL COMPUTATIONS

As an example of the use to which the computer may be put, it might be programmed to simulate functions equivalent to three-mode controllers having proportional, derivative and reset actions. Providion should also be made in this case for a small proportion of the controllers to operate in cascade as described in the following subsections. In addition to the example below, it should be kept in mind that other more eleborate algorithms and more complex control functions will be required of these machines. Such functions will probably not involve more than twice the memory and computational requirements outlined here.

12.1 The basic controller equation, if a position algorithm is being used, might be:

$$\text{Output} = K_p\, e + \Sigma\, K_R e + K_D(e_n - e_{n-1}) + K_1$$

where K_p, K_R and K_D are the proportional, integral, and derivative gains respectively. The error, e, should be calculated as:

$$e = \pm\ (\text{Set point} - \text{Controlled Variable})$$

The option of using either a plus or minus sign in the error calculation should be provided for each control loop. An adjustable midrange constant, K_1, should be included to provide for the possibility that proportional action only (i.e., K_R, K_D = 0) might be desired. The value of K_1 should be such that if the error, K_D and K_R were zero, the output would be midrange.

BEST AVAILABLE COPY

The summation shown in the controller calculation above should be taken over all previous iterations hence the current value of $\Sigma K_R e$ must be stored between iterations. Proper computation facts should be built into the controller calculations to prevent reset-- windup in either direction.

Should a velocity algorithm be desired, the basic controller equation becomes

$$\text{Output} = [K_p/\Delta t]\,(e_n - e_{n-1}) + (K_p/T_R)e_n$$

$$+ K_p/T_D(e_n - e_{n-2})$$

Here the output is a velocity signal which is transmitted to the value movement mechanism. Symbols are as follows:

Δt,	sampling interval for the variable in question. (See Item 2.14).
n,	relates to last sample obtained.
n-1,	relates to next to last sample obtained.
T_R,	reset control constant such that $K_p/T_R = K_R$
T_D,	derivative control constant such that $K_p/T_D = K_D$

Note that the midrange constant is not required.

The position algorithms would be used for an analog

output (see Paragraph 2.71) and with some implementations of pulse outputs (see Paragraph 2.72). The velocity algorithm would be used for most other pulse output systems. As mentioned elsewhere, pulse outputs are to be recommended.

12.2 Existence of a sudden "hard-over" signal condition on the input associated with a plant variable such as would indicate an input instrument failure should cause the individual controller calculation for that variable to be bypassed in order that its output remains at its last valid value. In other words, a failure in the input system will cause the process valve to remain in its last position until corrective action can be taken by the operator through the manual station on his console.

12.3 In order to achieve cascade control, provision should be made for the use of a controller output as the set point for another controller in approximately ten percent of the control loops. In the case of the following controller (i.e., a controller in a cascaded system whose output is the set point for another controller) the midrange constant, K_I, must be capable of being varied through the operator's console.

Control Loop Generation

Control loops should be constructed using modular function

blocks.

12.41 libraries of control algorithms, limit check algorithms, filter algorithms and output algorithms should be provided.

12.42 incremental control algorithms should be available.

12.43 the user should have the ability to construct algorithms beyond these supplied with the system.

12.5 Data Base

A structured file should be provided for specification or storage of control loop information such as:

12.51 input variables

12.52 messages

12.53 constants

12.54 control loop cascading specifications such that:

1. the operator may connect or disconnect cascade at any level
2. it should be possible to connect or disconnect cascade on request from a higher level
3. should the intermediate level of cascade

be disconnected, higher levels may optionally be disconnected as well

A means of easily accessing and modifying information in the file should be provided through the operator's I/O devices, the system I/O devices and high level programs.

13. INTERCOMPUTER COMMUNICATIONS

13.1 General

13.11 Hierarchical systems, vertically organized, will consist of "mother" and "daughter" systems. The "daughter" systems will be defined as those of higher dynamic requirements, and will usually be sensor-based, and smaller in size and cost than "mother" systems. "Mother" systems will ordinarily be management-oriented, but may also be involved in direct operation of production or testing equipment.

13.12 Parallel systems, horizontally organized, will consist of either "mothers" or "daughters", with approximately equal dynamic requirements.

13.2 System Discipline

13.21 System integrity is of primary importance so means must be provided for error detection. Failure of one element, such as a daughter component, should not cause a cascading of failures throughout the hierarchy. Local power failures or other failures must be noted by adjoining elements, even though normal data communications (or control of communications) may be one-way.

13.22 System generation methods will be used to structure the hierarchy, to determine the source

of control of data transmission for other than fault conditions.

13.23 Logic must be provided to synchronize the time of day between computers.

13.24 Illegal requests, such as attempts to write into protected areas, or to overflow buffer boundaries should be detected and prevented.

13.3 Hardware Capability

13.31 A minimum capability shall consist of ability to transmit or receive, half-duplex.

13.32 Capability shall be provided for at least binary and ISO, or binary and ASCII.

13.4 Software Capability

13.41 The same general procedures are to be used, whether the computers are local or remote from each other. System generation should accommodate timing restrictions due to distance and transmission modes for each channel used.

13.42 Software for data transmission should not distinguish between transmission of data and transmission of programs.

13.421 Capability should be provided for transmission of either random or sequential data.

13.43 Data transmission to other computers should be handled in a manner similar to methods used for handling other peripherals in the system.

13.44 The ability to output messages to a remote computers's peripherals is required. This type of request should take the form of standard I/O statements.

13.45 The option to either continue a program calling for data transfer or waiting until the transfer is completed should be provided.

13.5 Remote Task Selection

13.51 A provision must be made, for remote control of computer task selection. For example, data received may be intended for a file, or it may be a program or request for a program to be executed. System monitors must allow a means to examine a completed data transmission to determine the end-use of the data.

13.52 Transmission Control Characters

A standard control character or control character string format must be defined as part of the implementation of high level industrial computer languages.

14. FILE MANAGEMENT

A comprehensive system for management of the large amounts of data (including program files) should have the following characteristics:

14.1 The ability to define files to the operating system while causing their creation by means of a maintenance or utility program.

The following items should be available as defining parameters for creation of files:

14.11 Label: either a number or character string.

14.12 Device on which the file will exist

14.13 Type of file:

(1) Organization:

Sequential access, or

Random acess

(2) Record type:

Fixed length, or

Variable length

(3) File type:

Fixed length, or

Dynamically expandable

14.14 File Size

(1) number of records for a fixed length, or

(2) initial number of records, plus incremental number of records, plus maximum allowable number of expansions (increments) for dynamically

expandable files.

14.15 Record Size

(1) number of words/characters per record, for fixed length records, or

(2) maximum allowable number of words/characters, for variable length records,

(3) Blocking factor

14.16 File integrity

(1) Dynamically definable protection via "open" and "close" types of executable statements of application programs.

(a) "Open" file, with the following items as parameters:

File name or number

Definition of status as:

Reserved for exclusive use of "opening" program. (puts file into "protected" status for all other programs)

"WRITE" reserved for "opening" program (places file into "read-only" status for all other programs)

Unlimited access for any program

Error return, for the following conditions:

file nonexistance

file currently open with a conflicting status.

(b) "Close" file, with the following items as parameters:

File name or number

Error return conditions as follows:

File nonexistant

(2) Fixed definition of protection for life of file, with the following options:

(a) Read-only by any program except utility/maintenance system

(b) Unprotected - available for read/write use by any program.

14.2 The ability to remove or delete files by means of a maintenance or utility program, in such a manner that inadvertant deletion is difficult.

14.3 The ability to copy files from the device on which it resides to another device without destroying the original file, by means of a maintenance or utility program.

14.4 The ability to optimize location of storage allocated to files by means of a utility or maintenance program e.g. moving files, not stored contiguously, into continguity.

14.5 The ability to access logical records in a file by means of the following types of executable statements

in application programs, with return of error information to the application program (in such cases as: file non-existance, attempted access of a closed file, attempted write to a "read-only" file, access attempted beyond range of file, and attempted access of a "protected" file);

a. "READ" a logical record from the file into a variable or list of variables specified in the statement. (Subject to file integrity specfications)

b. "WRITE" a logical record from a variable or list of variables specified in the statements, to a file. (Subject to file integrity specifications)

BIBLIOGRAPHY

1. Anonymous, Minutes, Workshop on Standardization of Industrial Computer Languages, Purdue Laboratory for Applied Industrial Control, Purdue University, February 17-21, September 29-October 2, 1969; March 2-6, November 9-12, 1970; May 3-6, October 26-28, 1971.

2. Anonymous, "A Cross-Section of Opinion-Instrument Forecast for the 70's", Instrumentation Technology, 17, No. 1, 41-56 (January 1970).

3. Aronson, R. L., "Line-Sharing Systems in Plant Monitoring and Control", Control Engineering, 18, No. 1, 57-76 (January 1971).

4. Avizienis, Algirdas, et al., "Fault Tolerant Computing", Computer, 4, No. 1, 5-44 (January/February 1971).

5. Butler, J. L., "Comparative Criteria for Minicomputers", Instrumentation Technology, 17, No. 10, 67-82 (October 1970).

6. Elson, B. M., "Multiplexing in Jet Transports to Grow", Aviation Week and Space Technology, 89, No. 18, 157-161 (October 28, 1968), "Stored-Program Telementry Use Growing", Ibid, No. 16, 65-75 (October 14, 1968).

7. Keating, J. M., Private Communication, British Petroleum Company, London, England, June 16, 1966.

8. Kelly, E. A., "FORTRAN in Process Control: Standardizing Extensions", Instrumentation Technology, 17, No. 5, 47-53 (May 1970).

9. Lauher, V. A., Weinrich, D. W., and Underwood, R. K., "CRT Control Center for a Multi-Loop Process", Instrumentation Technology, 11, No. 9, 33-38 (September 1970).

10. Rosemblatt, Alfred, "TOPS Trails to Outer Planets Map a New Route to Reliability", Electronics, 43, No. 7, 108-115 (March 30, 1970).

11. Wiatrowski, C. A., "A Color-Television Graph Plotter for Digital Computers", Computer Design, 9, No. 4, 133-136 (April 1970).

12. Williams, T. J., Emulator, Simulator, and Translator Programs for the Small Digital Computers Necessary for Medical Laboratory Automation, Report 26, Purdue Laboratory for Applied Industrial Control, Purdue University Lafayette, Indiana, February 1970. Review Report, National Institutes of Health, Washington, D. C.

13. Williams, T. J., and Ryan, F. M., Progress in Direct Digital Control, Instrument Society of America, Pittsburgh, Pennsylvania (1969)

APPENDIX

QUESTIONS AND ANSWERS ON

DIGITAL COMPUTER BASED INDUSTRIAL CONTROL

As part of the original documents of the Users Workshop on Direct Digital Computer Control Systems held at Princeton, New Jersey, on April 3-4, 1963 and May 6-7, 1964 under the auspices of the Chemical and Petroleum Industries Division of the Instrument Society of America, a set of Questions and Answers on Direct Digital Computer Control were developed. The following is a listing with comments of those questions and answers given there which are considered to remain valid as of this date. The exact wording of the original questions has been altered somewhat in several cases to bring the questions up-to-date with present concepts and trends.

It should be noted that those Questions and Answers retained pertain almost exclusively to the characteristics of the plants being controlled or the desires of the operators for data presentation, etc. Practically all of those related to the computer equipment itself have been answered since by the accepted practices of the industry or are covered by specific items in the proposed new Guidelines which accompany this document.

In line with the backgrounds of the individuals who comprised the original Workshops most of these questions relate to the plants and problems of the Chemical and Petroleum Industries. Equivalent information should be developed for each of the other major industries using digitally based control systems.

QUESTION 1 -- What valve closing speeds are required for typical chemical and petroleum plant applications?

Is this a function of line size?

If so, what is a typical distribution of line sizes for a large chemical or petroleum processing plant?

ANSWER 1 -- Table VII gives some typical data for pneumatic valve closing speeds with and without volume boosters and positioners. Tables VIII and IX give some corresponding data for piston actuators and positioners manufactured by the Fisher Governor Company, Marshalltown, Iowa. Tables VIII and IX are derived from Fisher's Bulletin E-600, dated June 1961. Table VIII is from Page 12 of the Bulletin. Table IX is from Page 47.

It is the opinion of the industry group that the stroking speeds given in Column 5 of Table VII, "Pneumatic Positioner", are sufficiently fast to cover over 80% of the control applications in our industry. Those given in Column 3 of Table VII, "Kendall 3/8 Volume Booster", are sufficiently fast to cover over 95% of the possible control applications in the chemical and petroleum industries.

Of the remaining five percent, three percent can be readily handled by quick opening-quick closing, solenoid-type valves. The remaining two percent require control action

TABLE VII

STROKING SPEED IN SECONDS FOR CONTROL VALVES

VALVE SIZE	VALVE STROKE	Kendall 3/8" Volume Booster	Moore 61H Volume Booster	Pneumatic Positioner	Electro-Pneumatic Positioner	Electro-Pneumatic Transducer	Electro-Hydraulic Actuator
3/4-1"	1/2"	.3	.3	2.5	1.5	2.4	5.0 (1)
1-1/2"	3/4"	.5	.5	4.0	1.8	2.6	7.5 (1)
2"	1"	1.0	1.0	7.0	3.3	3.6	10.0 (1)
3"-4"	1-1/2'	1.5	1.5	13.0	6.3	7.5	15.0 (1)
6"	2"	2.5	2.5	20.0	11.4	14.0	4.0 (2)
8"-10"	2-1/2"	4.5	4.5	34.0	21.0	26.0	5.0 (2)
12"	3-1/2"	6.5	7.0	44.0	29.0	33.0	10.5 (3)
14"	4"	7.0	11.0	48.0	32.0	36.4	12.0 (3)

1. The above tabulation was extracted from the following sources:

   a. Mason-Neilan Control Valve Catalog No. 305 dated 8/59.
   b. Mason-Neilan Electro-Pneumatic Positioner Catalog Section 305-17 dated 9/59.
   c. Mason-Neilan Series 250 Electro-Hydraulic Actuator Temporary Bulletin No 306 dated 2/57.
   d. Askania (G.P.E. Controls) Folio D1 copyright 1959.

2. Inserting a Moore 61H volume booster with integral needle by-pass valve in the output of a positioner or transducer will result in improved stroking speed of the valve essentially as shown for the booster alone.

Notes:

(1) Askania Model 698 for valve strokes from 1/2" to 1-1/2" 500 lb. thrust.
(2) Askania Model 684-25 for valve strokes from 1/2" to 2-1/2" 1000 lb. thrust.
(3) Mason-Neilan Series 250 for valve strokes from 1/4" to 4" 3500 lb. thrust.

 BEST AVAILABLE COPY

TABLE VIII

PISTON ACTUATORS

SPECIFICATIONS

Actuator Size (1)	Cylinder Diameter, Inches	Effective Area, Square Inches	Yoke Boss Size, Inches	Stem Size, Inches	Available Stem Force 150 psi Supply	Available Stem Force 150 psi Supply	Allowable Stem Force, Pounds	Travel, Inches	Maximum Supply Pressure, psi (4)
23	4½	16.5	1¼	⅜	1800	—	1800	7/16-⅝	100
30	4½	16.5	2⅛	⅜	1500	—	1500	7/16-⅝	150
46	6⅛	28.25	2 13/16	½	2500	—	2500	7/16-1½	100
43	4½	16.5	2 13/16	½	1500	2200	2500	7/16-1½	150
60	8½	55.5	3 7/16	¾	5000	—	5000	7/16-2	100
63	4½	16.5	3 7/16	¾	1500	2200	5000	7/16-2	150
64	6⅛	28.25	3 7/16	¾	2500	3600	5000	7/16-2	150
80	10¾	88.5	5	1	5000	—	5000	⅜-3 (3)	100
86	8½	55.5	5	1	5000	8000	5000	¾-3 (3)	150
100	13	130.5	5	1¼	12000	—	12000	¾-3 (3)	100
130	17	227.5	5	1¼	12000(2)	—	12000	¾-3 (3)	60

Minimum Supply Pressure .....30 psi for all sizes.

Instrument Pressure Ranges.....3 to 15 psi, 6 to 30 psi, and others as required.

Bellows Pressure Rating.....50 psi. A high pressure bellows is available for pressures between 50 psi and 90 psi.

Temperature Limitation.....175°F. maximum.

(1) For the odd sizes of Series 470 actuators, namely 23, 43, 63, 64, and 86, the first digit indicates the yoke boss size and the second digit indicates the cylinder size. For example, a Size 64 would indicate a 3 7/16" boss and a 60 cylinder.

(2) Based on 60 psi supply pressure.

(3) These actuators have 4" travel available but travel is limited to 3" when mounted on Fisher valve bodies.

(4) Based on allowable stem force. All actuators, except Size 130, can handle pressures up to 150 psi. The Size 130 is limited to 60 psi due to the strength of the yoke even though the cylinder is designed to handle 150 psi.

CONSTRUCTION MATERIALS

Cylinder

Cylinder and Piston .....Aluminum
Piston Rod and Extension.....Stainless Steel
Cylinder Seal Bushings.....Brass
Seal Rings.....Synthetic Rubber
Yoke .....High-Tensile Iron

Positioner

Base, Cover, and Beam .....Aluminum Die Casting
Bellows .....Brass
Relay Body .....Zinc Die Casting
Relay Nozzle .....Stainless Steel

PERFORMANCE DATA

Air Consumption (Static).....20 scfh with 100 psi supply pressure.

Hysteresis (Maximum).....0.15% of total stroke or instrument pressure span.

Repeatability.....0.03% of total stroke or instrument pressure span.

Resolution Sensitivity.....0.02% of instrument pressure span.

Frequency Response.....See Figure 1-28.

Maximum Stroking Speed

(Based on supply pressure indicated)

Actuator Size	30	40	60	80	100	130
Supply Pressure, psi	100	100	100	100	100	60
Speed, in/sec	4.0	2.66	1.3	0.8	0.59	0.33

Stroking speeds listed do not apply to Types 471, 472, 473 and 474

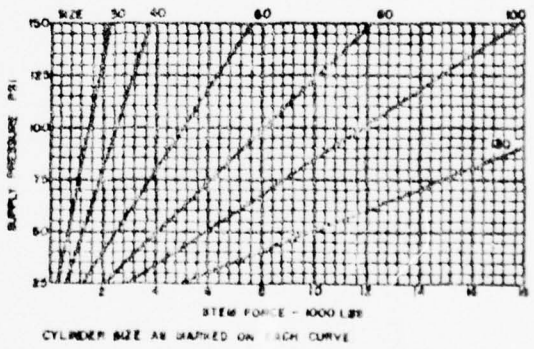


Figure 8 Stem force versus supply pressure for Series 470 actuators equipped with Type 1070 positioner. Dwg. AH7648

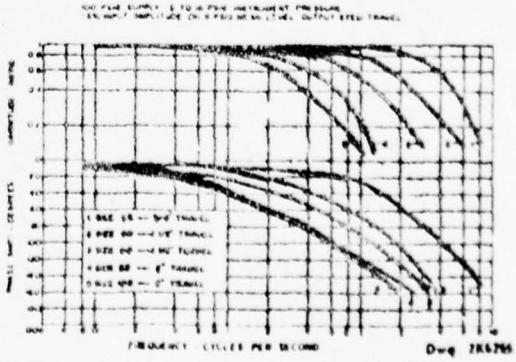


Figure 9 Frequency response curves for Type 470 actuators.

BEST AVAILABLE COPY

TABLE IX

SERIES 3560 V/P VALVE POSITIONER

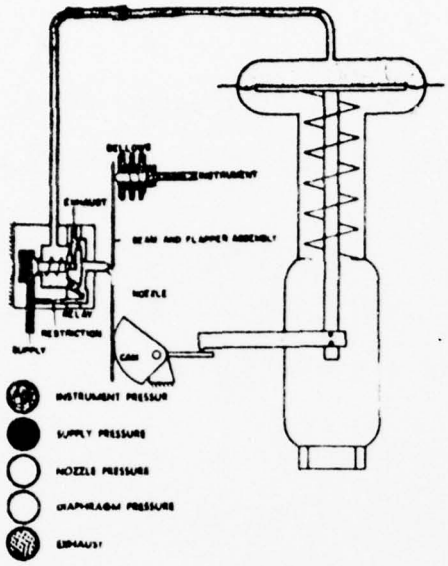


Figure 10 Type 3560 schematic. Dwg. CR3896

Reverse acting positioners operate in a similar manner except that as the instrument pressure increases, the diaphragm case pressure is decreased. Conversely, a decreasing instrument pressure causes an increase in the diaphragm case pressure.

ADJUSTMENTS

Valve travel is easily adjustable for any travel between ¼" and 3" (as required by actuator size) by simply moving the flapper arm along the beam. The beam is labeled to indicate that movement of the flapper arm away from the cam increases the valve travel. The relay nozzle is adjustable and is used to set the starting point of valve travel at the desired value of instrument pressure. These two simple adjustments make the Series 3560 readily adaptable for split range operation.

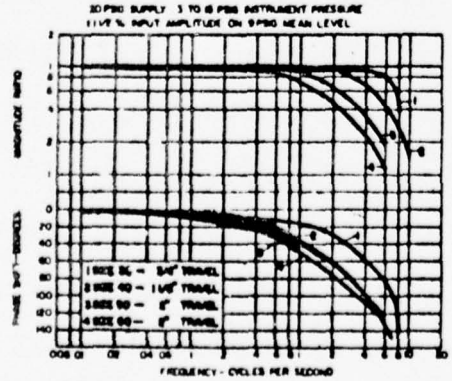


Figure 11 Frequency response curves for Type 3560 on Type 657 diaphragm actuators. Dwg. BK6959

The V/P is reversible without the use of additional parts. For direct action (increasing instrument pressure increases output pressure), the flapper arm is positioned on the bellows quadrant of the beam. For reverse action (increasing instrument pressure decreases output pressure), the flapper arm is placed on the opposite quadrant of the beam.

SPECIFICATIONS

Maximum Supply Pressure ..... 50 psi
Pressure Rating of Bellows ..... 35 psi
Nozzle Orifice ..... 0.040" diameter
Fixed Restriction Orifice ..... 0.018" diameter
Pressure Gauges .... 1½" diameter weatherproof gauges with 0 to 30 or 0 to 60 psi range
Pressure Connections ..... All ¼" NPT
Dimensions ..... 8¾" high and 6⅝" wide (with pressure gauges)

Available Range Springs*

Instrument Pressure Range, Psi	Range Spring Dwg. No.	Spring Color Code
3 to 15	1J9678	Cadmium Plate
6 to 30	1J9680	Red

*Range springs for other instrument pressure ranges will be available as required.

PERFORMANCE DATA

The following data has been taken from tests performed in Fisher's Research Laboratory on Type 657 diaphragm actuators equipped with a Type 3560 V/P. The instrument pressure range was 3 to 15 psi and the supply pressure was 20 psi. Actuator sizes and travels are indicated in the tabulation.

Actuator Size	30	46	56	60
Travel, Inches	¾	1½	2	2
Stroking Speed, in/sec	0.5	0.4	0.3	0.2
Open Loop Gain (with linear cam) (1)	68	86	86	86
Linearity (with linear cam) (Instrument Pressure vs. Valve Travel)	Within ± 1%			
Resolution Sensitivity (2)	Less than 0.05% of Instrument Pressure Span			
Repeatability (3)	Less than 0.03% of Full Travel			
Steady State Air Consumption, scfh	10			

(1) The ratio of output change per unit input change through a system when the control loop is open.

(2) Resolution sensitivity is the minimum amount of change in the instrument pressure that will create a measurable movement of the valve stem.

(3) Repeatability is the ability of the V/P to reposition the valve stem for a given value of instrument pressure.

but must have a closing speed of 0.6 second or less regardless of the size valve involved.

Table X presents distribution data for the range of valve sizes used in one of our industry's larger chemical plants.

Nothing has occurred recently in the design of plant control systems or in the development of new components which promises to appreciably change this data for continuous process plants.

QUESTION 2 (Previous Question 3) What shoud be the precision of the constants K_R (Integral Gain) and K_p (Proportional Gain) of Item 2.61, of the Guidelines?

What limits or range of variation should be placed on these constants?

Who should set and adjust these constants in the machine?

ANSWER 2 -- The desired range and precision of setting of the gains of the control modes of the direct digital control computer are as follows:

PROPORTIONAL - Settings of the proportional mode on conventional pneumatic and electronic process control hardware generally range from 10 percent proportional band to 500 percent proportional band. Those few which are outside this range (up to 1000 percent) can be handled by special programming steps. The given proportional bands correspond approximately to loop gains ranging from 10.00 to 0.20.

TABLE X

DISTRIBUTION OF VALVES ACCORDING TO LINE SIZE

PETROCHEMICAL PLANT APPLICATIONS

Line Size in Inches	Percentage
3/4	22.3
1	7.4
1-1/2	21.1
2	22.1
3	8.4
4	6.4
6	6.8
8	3.1
10	0.6
12	0.8
14	0.2
18	0.4
20	0.4
	100.0

In making settings of mode gains, resolution in making the setting itself is far more important than the actual setting involved. For this reason, four digit adjustment is required on the computer. Thus a digital adjustment to 0.01 would be desirable.

Approximately 90% of current valve applications would use the 10-500 percent proportional band given above. Approximately 10% would be outside this range, requiring gains to 1 percent proportional band in one extreme or up to 1000 percent in the other.

INTEGRAL - Integral or reset mode gains are commonly expressed as the amount of time required for the development of a correction equivalent to a proportional band of 100% or a proportional gain of 1.0. Such times commonly range from 10 seconds to 15 minutes. Again, those few which are outside this range (up to 60 minutes) can be handled by special programming. Another common method of writing these gains is to give the number of times an integral mode correction, equivalent in size to the proportional band mode correction of the instrument, is generated each minute. In this representation a reset time of 15 seconds would be roughly equivalent to 4 resets/minute; 0.1 resets/minute would give a reset time of ten minutes, etc. These given reset times correspond to integral mode gains of from 1.000 to 0.001.

As was mentioned under proportional above, resolution in making the setting is more important that the initial accuracy of making the setting. A digital adjustment to 0.001 would be very desirable.

Again about 90% of the applications would be in the range indicated. Some very few cases may require reset down to 1 second. Approximately 10% of the cases will require reset times of from 15 minutes to 60 minutes.

DERIVATIVE - Derivative or rate mode of control was not called for in our original specification and we at this time see no compelling reason to include it in the example problem there. However, the consensus of the Users Group in discussing direct digital control was that derivative control would be called for in up to 20% of the loopsto be included in a system.

To avoid the problem of system noise involved in determining a system derivative digitally, an analog derivative might be used. While this would necessitate two separate inputs for the variable, they may be necessary for smooth control.

The magnitude of derivative action imposed in a controller is usually expressed as a "rate time" or the difference between the time required to complete a given correction by proportional action alone and that necessary for proportional plus rate control taken together. This number expressed in minutes and multiplied by the proportional gain equals the true derivative gain of a noninteracting derivative mode. Rate times normally vary between ten seconds and fifteen minutes. Since the longer times are used only with small proportional gain systems, this results in equivalent derivative gains of between 10 and 0.01. Again, a four digit setting capability would be desirable for purposes of resolution as mentioned above.

Since these concepts are foreign to the usual interpretation of gains in servo loops as taught to electrical engineering students, the following references are given for further study as desired:

1. Eckman, D. P., "Automatic Process Control", John Wiley and Sons, Inc., New York, New York, 1958, Chapter 3, pp. 59-77.

2. Holzbock, W. G., "Automatic Control: Principles and Practice", Reinhold Publishing Corporation, New York, New York, 1958, Chapter 4, pp. 25-60.

3. Tucker, G. K., and Wills, G. M., "A Simplified Technique of Control Systems Engineering", Minneapolis-Honeywell Regulator Company, Philadelphia, Pennsylvania, 1958, Appendix F., pp. 235-256.

The adjustment of controller mode constants in chemical process control systems is a duty which is permitted of only a very few individuals. Normally, only the plant instrument engineer or the lead instrument mechanics are permitted to touch these settings. Presumably the same restrictions would apply in the case of direct digital control computers since the same problems of plant stability would exist regardless of how control activation were actually initiated.

QUESTION 3 (Previous QUESTION 5) -- Are engineering units required on any digital logs which might be used by the operator for plant monitoring?

Are exact temperatures required on logs or might some simpler representation suffice for operator use?

ANSWER 3 -- It is common practice in the process industries to present flow rates and vessel levels as percentages of some predetermined maximum value on recorder charts and logs. On the other hand, because of their greater importance to plant safety, temperatures and pressures are usually presented as their true values in degrees Fahrenheit and pounds per square inch respectively.

In order to facilitate operator familiarization with the new type of control system which would be involved with direct digital control, it is highly desirable that this system of presentation of plant data be preserved.

The statements just given should be taken as a clarification and expansion of Item 1.234, and of Item 2.32, of the Functional Guidelines on Industrial Control Computer Systems.

It should be noted that there is a sharp difference of opinion in the process industries concerning whether digital logs or analog strip chart recorders should be used for presentation of the majority of plant data to the operators. It appears that both options may be required of the vendor.

QUESTION 4 (Previous QUESTION 6) -- In order to prevent reset windup in the computer control computations, should any special external signals be provided to control the reset calculation, or would internal testing of the reset calculation be sufficient?

ANSWER 4 -- Reset windup is the continued buildup of an integral control correction while the controller is receiving an input error but is not making an actual correction of the error source (as when the controller is in an open-loop configuration). This can be a very annoying and troublesome condition during plant startup, when taking control loops off and on-line such as for repairs or replacement, when a severe upset occurs in the process, or during the operation of batch processes. It can be controlled or corrected in several ways. Among these are:

1. Establish a maximum limit which the controller output can assume, i.e., allow the integral control mode to wind up to only a relatively small value.

2. When windup occurs, adjust the process loop set point to such a value that a large error of the opposite sign is sensed by the controller thus integrating out the windup signal.

3. Whenever putting a control loop on or off line set the value of all integral mode outputs to zero and require a recomputation of any necessary integral correction.

4. Whenever putting a control loop on or off line, make momentary use of a very large integral constant to cause the integral control loop output to quickly assume its correct value.

While Point 3 above is often unsatisfactory because an appreciable integral correction may be the legitimate controller output at the moment in question it has become the most used of the alternatives presented. Point 4 may impose the upset of a large integral correction on the system if not used carefully but does hasten its removal.

A combination of Points 1 and 2 would thus be one of the best remedies. Since control loops are normally put on line one at a time, the operator could personally make the adjustments called for in Point 2. Therefore, while an automatic device to handle this condition would be a convenience, its monetary value in terms of a justification would probably be quite small.

Item 2.61, of the Guidelines requires that factors be built into the control computations to prevent reset-windup. The limiting method discussed under Point 1 would be the minimum acceptable solution to the problem.

As mentioned in Item 2.86, the fast reset of Point 4 above may be employed with the manual back-up system provided it is used just prior to putting a loop on line and is disconnected before the automatic control loop is actually connected into

the process. It is estimated that the reset-windup will be a problem in 20% of the cases where the reset control mode is used.

QUESTION 5 (Previous QUESTION 9) -- Item 3.1, of the Guidelines calls for an on-stream availability of 99.95% of plant operating time. How does this translate into mean-time-between-failures and allowed down-time at failure?

Just what items of plant and control equipment must be included in the system for which the availability above is to be determined? If such high reliabilities and availabilities can be obtained, what can be done about maintaining plant operator and computer maintenance personnel proficiency?

ANSWER 5 -- Extremely high availabilities of computer control equipment are necessary to prevent frequent and costly shutdowns of chemical plants. One unscheduled shutdown of the plant can cost the process company in lost production many times any potential capital or operating

savings which the computer can give over conventional control systems. Therefore, such shutdowns must be reduced to an absolute minimum if these new control systems are to be accepted generally throughout the process industries.

Figure 12 represents a possible compromise between mean time of failure and system downtime which would be acceptable to us. It is the consensus of opinion of our people that 99.95+% availability is unquestionably the goal which we all desire.

Parts of the system to be included in assessing availability or reliability are as given below. These are essentially all of those items whose failure could result in an interruption of operation of a large portion of the control hardware, i.e., up to 1/4 or 1/3 of the total number of control points or greater.

1. Computer Mainframe and Associated Parts.
2. Input Multiplexer and Associated Circuitry.
3. Analog to Digital Converter if Multiplexed.
4. Output Multiplexer and Associated Circuitry.
5. Digital to Analog Converter if Multiplexed.
6. Alarm Detection, Indication, and Printout Devices if Common to a Large Portion of the System.

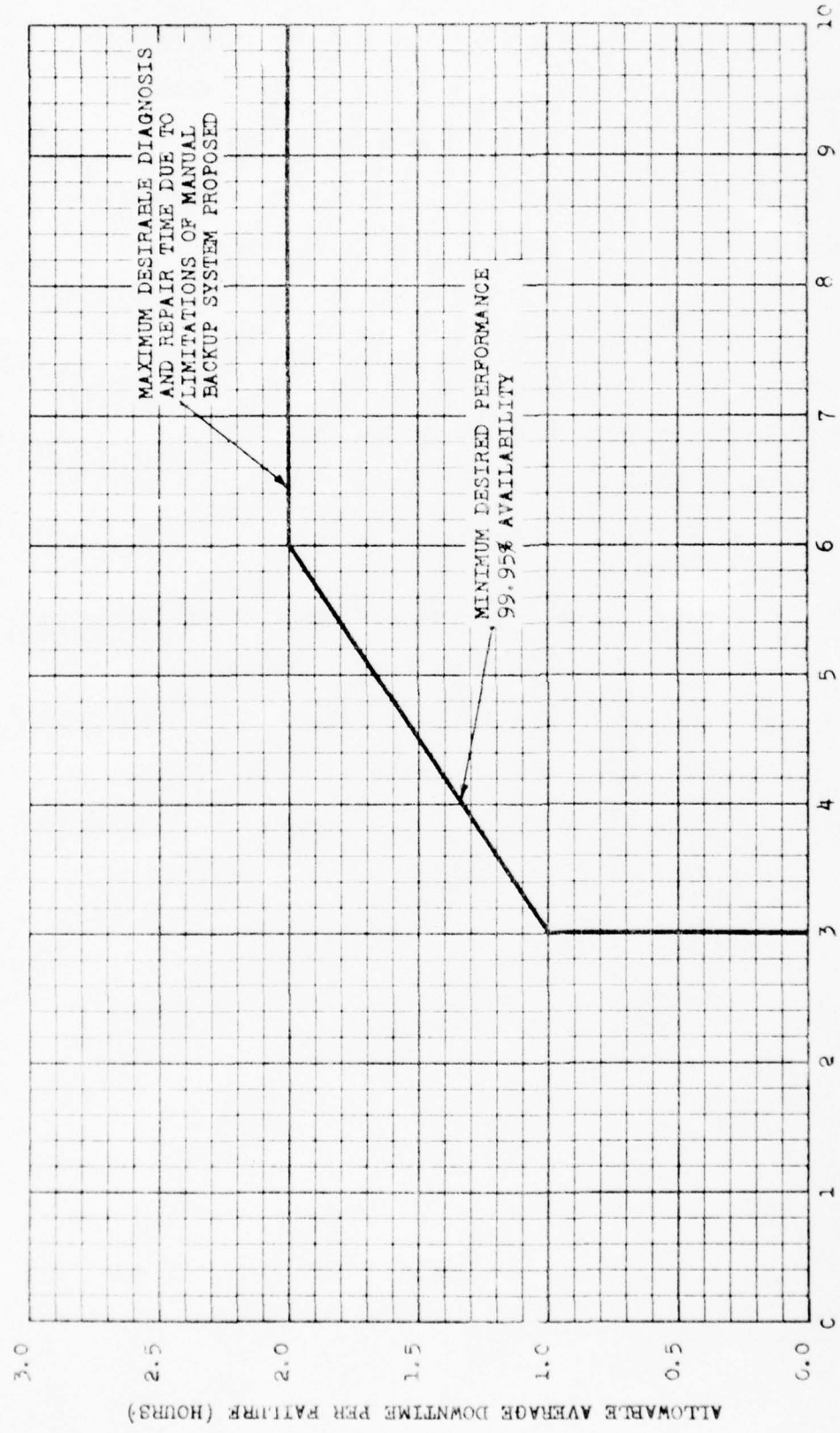


FIGURE 12

RELATIONSHIP OF MEAN TIME BETWEEN FAILURES AND ALLOWABLE REPAIR TIME FOR DIRECT DIGITAL CONTROL COMPUTERS

Those items which definitely would not be included in the equipment whose availability is being judged would be the following. Again, these are those items normally involved with the operation of only a single control loop and those items whose failure would affect only that loop:

1. Individual transducers.
2. Individual transmitters.
3. Individual valves or valve actuators.
4. Individual alarm indicators used for signal of only one or a small number of points.

It should be noted also that the extremely repetitive nature of the operation of a direct control computer would make "dropped bit" errors essentially negligible in their effect upon plant control. This is especially true for the integral mode of control. Such errors would thus be considered in figuring reliability only when they begin to occur so frequently as to impair control accuracy.

Maintenance of competence on the part of operating and repair personnel is a very serious problem. This is especially so as system availabilities become very high and required performance on the part of plant personnel becomes less frequent. It is the consensus of opinion of our people that we would maintain proficiency by coming "off-line" for regularly scheduled short periods of time

for training. Such training would include manual control of plant changes and could include preventive maintenance of the computer. Such periods would not be charged against system availability records.

QUESTION 6 (Previous QUESTION 10) -- What time period is normally required for a safe crash shut-down of a chemical plant?

What is the philosophy of shut-down vs. stagger along operation in event of direct control computer failure?

What type of units would we want to shut-down?

What type of units would we want to keep going?

ANSWER 6 -- Required shut down periods for chemical plants will vary greatly depending upon the type of unit involved and the possibility of damage to plant units due to thermal and mechanical stresses and corrosion or the possibility of the danger of explosive conditions existing.

Particularly sensitive are furnaces, fired reboilers, and other such units which must have their temperatures reduced slowly to prevent thermal cracking of firewalls, etc. Equally sensitive but for the opposite reason are refrigerated condensers, etc., of low temperature columns. Here temperatures must be raised slowly to prevent undue mechanical and thermal stresses. A period of several hours to days may be required especially for the furnaces.

Another area requiring care is the handling of materials susceptible to polymerization or decomposition when subject to abrupt temperature changes or exposure to air or moisture. Finally, there is the problem of maintaining control of reactors handling highly exothermic reactions.

In each of the last two items above, close control of unit temperatures, purge flows, etc. for a definite period of time is involved. However, there is usually a considerably shorter period than for the furnaces mentioned earlier.

Where such special conditions do not exist, crash shutdowns can be handled in a very short period of time.

It should, however, be emphasized here that it is generally process plant policy in continuous process plants to continue plant operation whenever possible unless a condition very dangerous to life or property exists. This is because of the difficulty and very long time periods required to start up many units and to the very large losses in production involved.

Thus, only highly exothermic or very short time constant reactors or other similar units would normally be shut down. Such items as distillation and absorption trains which require very long start-up periods would be kept operating if at all possible.

QUESTION 7 (Previous QUESTION 11) -- What bit count is required internally to handle the smallest feasible integral constant?

ANSWER 7 -- In the answer to Question 3 above we stated that the smallest required integral constant was 0.001/sec. Since an accuracy of four digits was asked for on input variable readings, the minimum integral correction involved would be:

(0.001/sec.)(0.0001 units) = 0.0000001 units/sec.

Thus seven digits would be sufficient to compute the slowest integral correction desired.

QUESTION 8 (Previous QUESTION 14) -- Is an Alarm Unit a must item or simply a desirable one?

What is your evaluation of the method of indicating alarms by a pre-recorded "voice output" annunciator system?

ANSWER 14 -- An Alarm Unit is an absolute necessity in a plant control system. It must give a distinctive and unmistakable indication of trouble as soon as possible after it occurs and should in addition pinpoint the source and nature of this trouble. Pre-recorded "voice output" annunciators offer a distinctive type of announcement particularly if in a voice which is incongruous with the surroundings (such as "southern female"). However, such a method of indication would not have sufficient attraction over the present horn and bell type systems to justify the outlay of large sums of money.

It should be noted that, regardless of what method of indication is used, it should be repeated until acknowledged by the operator to confirm that he is aware of the presence and nature of the plant trouble.

QUESTION 9 (Previous QUESTION 16) -- Do we still believe that the sampling speeds listed in Item 2.14 of the Guidelines are adequate for chemical plant control?

Have we uncovered any special cases which might require more frequent sampling?

ANSWER 9 -- An extensive study of plant requirements has uncovered only a very few cases where a sampling speed higher than those mentioned in Item 2.14 of the Guidelines would be desirable. One of these is in the pressure monitoring of highly exothermic gas reactors. Here a sampling speed of up to 5 readings per point would be required. Others might require speeds up to 10 reading per second per point. However, these and similar units are so few in number in chemical plants that they could be included as multiple points under the regular sampling requirements listed in the Guidelines.

QUESTION 10 (Previous QUESTION 17) -- What is the present distribution as a percentage of the total , of the various types of control loops in our plants?

ANSWER 10 -- Recently a study was made of the percentage distribution of various types of actual control loops in some of our largest plants. The following approximate numbers were obtained for four such plants:

	(1)	(2)	(3)	(4)
Flow -	63	40	30	38%
Pressure -	10	18	26	15%
Level -	10	25	25	20%
Temperature -	15	17	18	25%
Analysis -	2	0	1	2%

Note: Between 6-10% of the total loops are arranged in cascades for final control actuation.

The distribution of sensors for process variables as read by the computer is somewhat different from this because of sensor points required for alarms, logging, etc. A distribution of total read-in points required for input to a computer system for another of our large plants is as follows:

Flow	40%
Temperature	33%
Level	11%
Composition	9%
Pressure	7%

QUESTION 11 (Previous QUESTION 25) -- Can we better define the concept of a control loop?

Possible additional information which should be included is as follows:

a) How many input variables would we normally consider as a maximum to be involved with any one valve actuator or defined control loop.

b) What is the ratio of alarms to defined loops?

c) What is the ratio of logging outputs to defined loops?

d) What is the ratio of potential recording requirements to the number of defined loops?

ANSWER 11 -- Answering this question properly requires a more specific definition of the term control loop than has previously appeared in the Guidelines or related publications. This definition is as follows:

"A Control loop shall be defined as each and every separate combination of inputs and stored quantities, of arithmetic operations, and of possible output procedures which together result in the determination of the existance of an error or deviation from some established process value and which carry out the computation and activation of a resulting control or corrective action which latter is designed so as to ultimately bring the process value back to its correct value or to compensate in the process for the detected deviation. In this context each separate cascade control computation, each feedforward control procedure, each ratio control determination, as well as each conventional control algorithm application which results in an actual valve or other final control element movement shall be counted as a control loop."

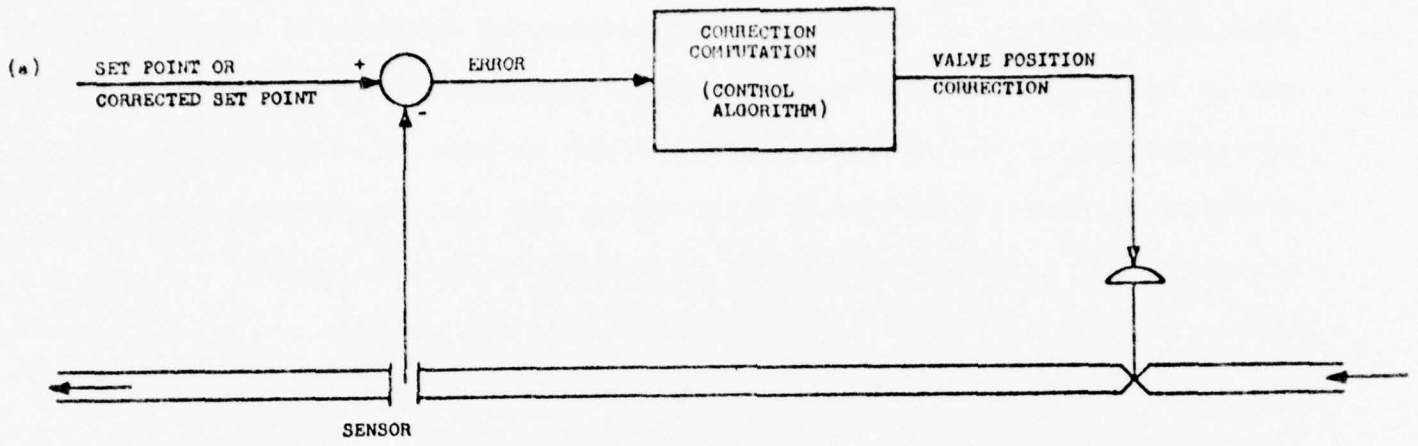
Figure 13 illustrates some possible examples of control loops according to the above definition.

a) Input variables per actuator or defined control loop.

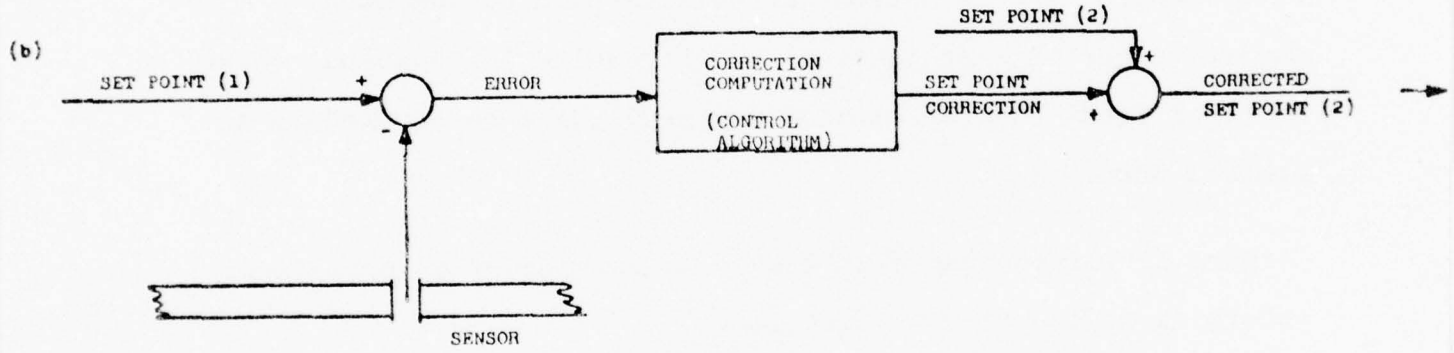
(1) At present the distribution of required inputs (those actually used in control computations) per control loop averages 1.5 to 1 for chemical plants and petroleum refineries and up to 2 to 1 for cement plants and steel mills. It is felt that the ratio will increase steadily in the next few years and settle at 2 to 1 for all types of plants.

FIGURE 13

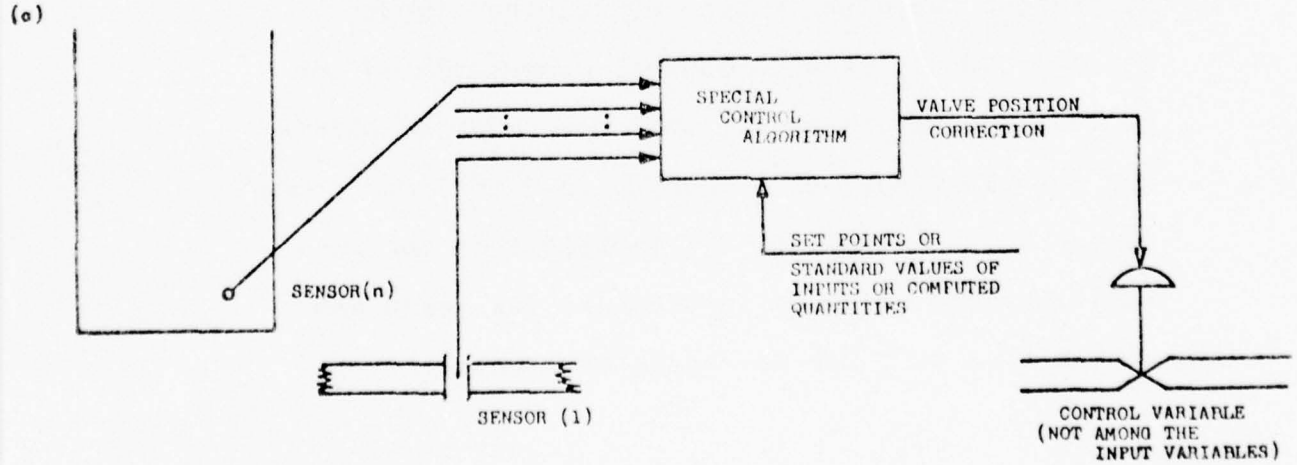
SOME EXAMPLES OF CONTROL LOOPS PER
THE DEFINITION OF QUESTION 25



BASIC CONTROL LOOP

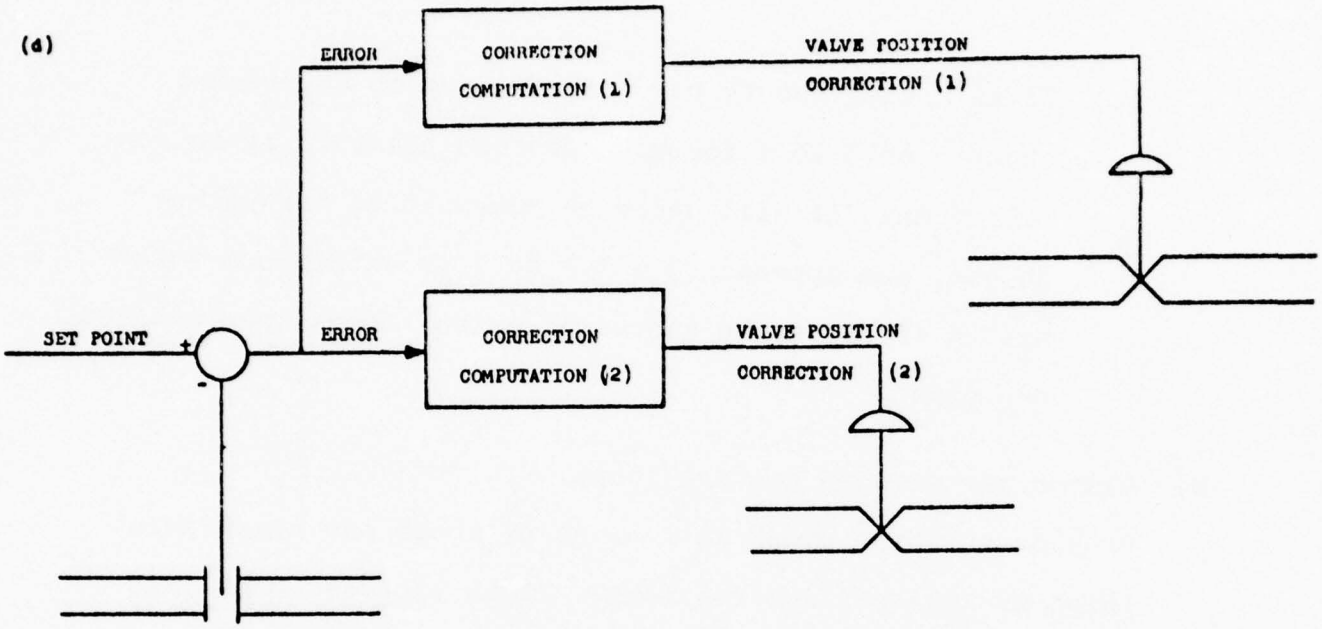


ELEMENTARY CASCADE LOOP

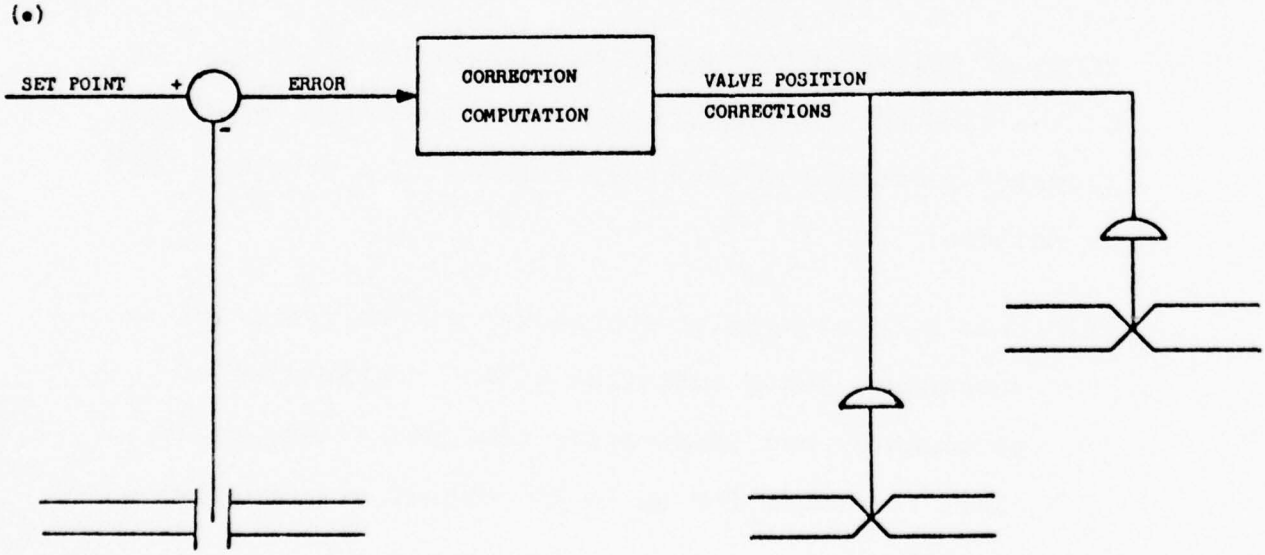


GENERAL FORM OF A SPECIAL CONTROL FUNCTION
SUCH AS FEEDFORWARD AND ADAPTIVE CONTROL
(COUNTS AS ONE CONTROL LOOP REGARDLESS OF
NUMBER OF INPUTS)

FIGURE 13 cont.



MULTIPLE OUTPUT COMPUTATION - SINGLE INPUT



MULTIPLE VALVES - SINGLE COMPUTATIONS - SINGLE INPUT

UNDER THE CRITERION OF THE DEFINITION OF ANSWER 25:

(d) WOULD CLASSIFY AS TWO LOOPS SINCE TWO SEPARATE CONTROL COMPUTATIONS ARE MADE.

(e) WOULD CLASSIFY AS ONE LOOP.

(2) Total analog inputs per control loop is at present about 4 or 5 to 1 for most process plants. It is our belief that it will decrease somewhat in the years to come and approach 3 - 3.5 to 1 as extraneous information is discarded and more control loops are involved per plant.

b) Alarms per defined control loops.

Present practice shows an average of about one alarm point (high or low) per control loop. It is expected that this number will increase to about 1.2 to 1 as more sophisticated monitoring systems are added to future control installations.

c) Ratio of logged and/or recorded outputs per defined loop.

On the subject of logging and analog recording, process industry personnel are sharply divided into several camps as follows:

(1) Some wish no logging whatsoever either permanent or temporary, being satisfied with a combination of permanently and temporarily assigned analog strip chart recording for up to 50 percent of the input variables.

(2) A second group would rely mainly on digital recording partly temporary for operator's aid, and partly permanent for recording process data for historical

purposes and later analysis. They would have only a minimum of analog recording, this to be temporarily assigned and used mainly for analog backup purposes.

A vote among the attendee companies regarding their feelings toward one or the other was as follows:

(1) All companies attending (25)

Would Not Use Record Logs But Rely on Analog Recordings	Prefer Digitally Logged Records	Abstain or No Experience
10	10	5

(2) Of those with on-line computer experience (13)

Would Not Use Record Logs But Rely on Analog Recordings	Prefer Digitally Logged Records
5	8

Of those who voted for logging the consensus of opinion was that there would probably be a logged output for almost every input, i.e., a total of about 3 or 4 to 1 in relation to control loops.

QUESTION 12 (Previous QUESTION 26) -- What is the maximum number of loops which should be centralized in one operator's console?

What is the maximum number of loops which should be centralized in one digital computer system (this should be answered both for single computer systems and for possible dual computer systems)?

What is the maximum number of loops which should be assigned to any one operator (in terms of his performance during manual control periods)?

ANSWER 12 -- Using control loops as defined above the following possible distributions were proposed:

a) Loops per computer -- 20 to 150, i.e. put the complete plant on one computer up to the 150 loop limit.

b) Loops per operator -- 20 to 50. One operator could handle a complete small plant (up to 50 loops). However, for larger plants, more operators should be used for backup control with the limit on required work being about 50 loops per operator.

c) Loops per operator's console -- 20 to 150. This is a function of human engineering in combination with the rules expressed above. One console could be used for a 150 loop plant provided 3 or 4 operators could work at it without interfering with each other. Otherwise several smaller consoles should be provided with the loops divided conveniently according to the plant units involved.

QUESTION 13 (Previous QUESTION 27) -- What percentage of the loops in an up-to-date petrochemical plant would have cascade control involved in their operation?

Do we expect this number to increase in the future?

What is the potential for the use of multiple cascade?

Which of the many special analog computer type control algorithms do we expect to be included in the capability of the direct digital control computer?

Can we assign any ratio of such functions to the number of standard control functions in the plant?

ANSWER 13 -- The present number of cascade loops in a modern chemical or petroleum plant comprises 10-15% of the total defined control loops. It was the unanimous opinion of the Workshop attendees that this would increase rapidly in the future to at least 25% of the defined loops. Multiple cascades would be an important percentage of the cascaded loops -- perhaps as much as one-third.

The attendees were emphatic in their opinions that the many special analog algorithms would be a very important factor in future direct digital computer control. They felt that most of the published algorithms would be useful and provision for their possible use should be included in the capabilities of the computer. It was the opinion of the group that such algorithms would also increase rapidly and would also approach 25% of the defined loops.

The net result of the above would be that about half of the future plant control loops would be involved in one way or another with cascade or other advanced control functions.

QUESTION 14 (Previous QUESTION 31) -- When the direct digital control computer is used in conjunction with an optimization computer, how much data must be transmitted to the optimization computer and how often should it be transmitted? (Assume a "typical" petrochemical plant of 100 control loops and 350 inputs total.)

ANSWER 14 -- Enough data must be transmitted to assure an accurate and up-to-date optimization function for the plant. A suggested "transmission load" would be as follows:

1. Transmit a complete set of plant input data to the optimization computer once during each five to ten minute period, i.e., 350, ten bit, values. Length of this period would be based on the optimization requirements of the subsidiary plant.
2. Return a complete group of new set points to the direct control computer once during each basic optimization period, i.e., 100, ten bit, values.
3. In order to assure time coincidence of the data and set points, all transmission (in both directions) should occur within the same one minute period within the longer block and should involve currently read and calculated values.

The net result of the above is that the optimization computer by means of the data links could service up to 10 separate direct control computers if its own internal memory capacity and operating speed were sufficient to handle this computational load. If fewer subsidiary computers are involved, off-line scientific or accounting work could be carried out.

QUESTION 15 (Previous QUESTION 35) -- What are the latest opinions (May 1964) of the Users concerning requirements for operator's consoles and manual backup systems?

ANSWER 15 -- Answers to this question were given by several companies who had made actual studies of this subject. They are as follows:

1. First Chemical Company

   a) An automatic/manual transfer should be possible with each loop individually as well as a mass transfer on detected computer failure. One suggestion is to use an individual three-way toggle switch for each loop. When up, the system would be computer operated; when neutral it would be disconnected (i.e., system frozen); when in the down position (springloaded) it would be possible to adjust the valve position from an adjacent knob or switch or push button.

b) A graphic or semigraphic display of the plant should be included.

c) Analog display of process variables should be switchable and multiplexed approximately 50/1. Display should be in the form of a normalized needle with the normal operation region centered. A color coded dial instead of a numerical dial would be helpful if feasible.

d) Analog display of valve position should also be switchable and multiplexed approximately 50/1. It should present valve position as percent of full stroke. An associated push button would be helpful (as long as the push button is pressed a continuous presentation of the valve position would be given, when released the last value would be preserved).

e) All stored constants and all measured variables should be accessible to a digital display. It would be desirable to have six separate digital displays (3 digits each in percent of full scale), if not too costly, per set. One set would be used for each 50 loops. The six displays should give all six values pertaining to any one loop at the same time (variable value, set point, valve position, three separate gains). Loop selection should be by digit switches.

f) A SCAM type alarm indication would be used.

g) Permanently assigned analog recording would be used for about 15% of the variables.

h) Temporary (switchable or pluggable) analog recording would be used for an additional 5% of the variables.

i) No digital logging would be used. All permanent records would be made by associated supervisory computers.

2. Second Chemical Company

Essentially agreed with the above views except for the following changes:

b) No graphic or semigraphic display would be required.

e) Digital display should be to 4 digits. Displays should be in engineering units. Three separate displays per set would be sufficient instead of the six requested above.

g) Permanently assigned analog recording - 5%.

h) Temporarily assigned analog recording - 15%.

i) Logging would be an integral part of the direct control system.

3. Third Chemical Company

Would keep console display to minimum, use one set of four indicators (percent full scale needle for backup, digital for normal operation) for all loops. Have these indicators

show point number, set point, variable value, and valve position simultaneously under control of single rotary switch system.

The attendees also gave some thoughts which they had on possible desirable future developments for operator's consoles:

a) Plan on using a data recording and playback system such as by disk file and scope display in place of analog strip chart recording. Recall should be as follows:

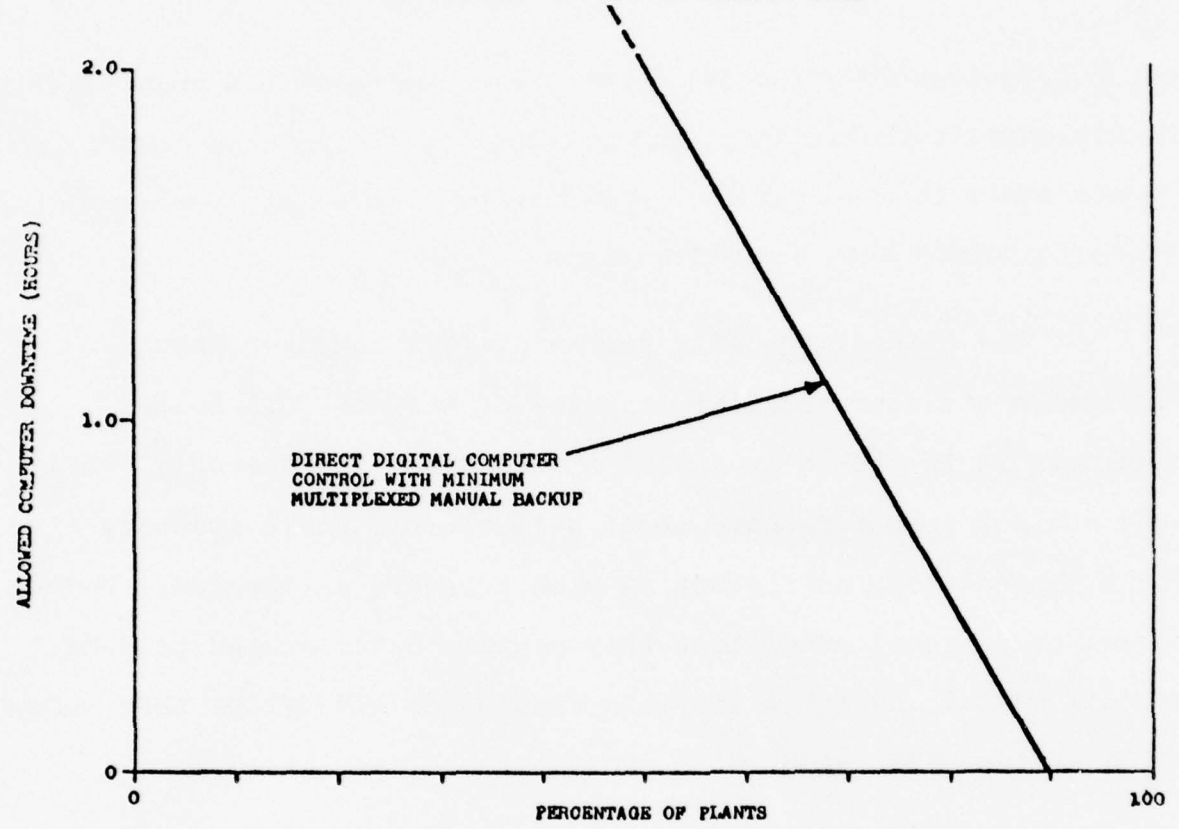
   (1) Long time constant system information - variables sampled each several minutes, allow recall up to 24 hours. (Will comprise about 20% of loops.)

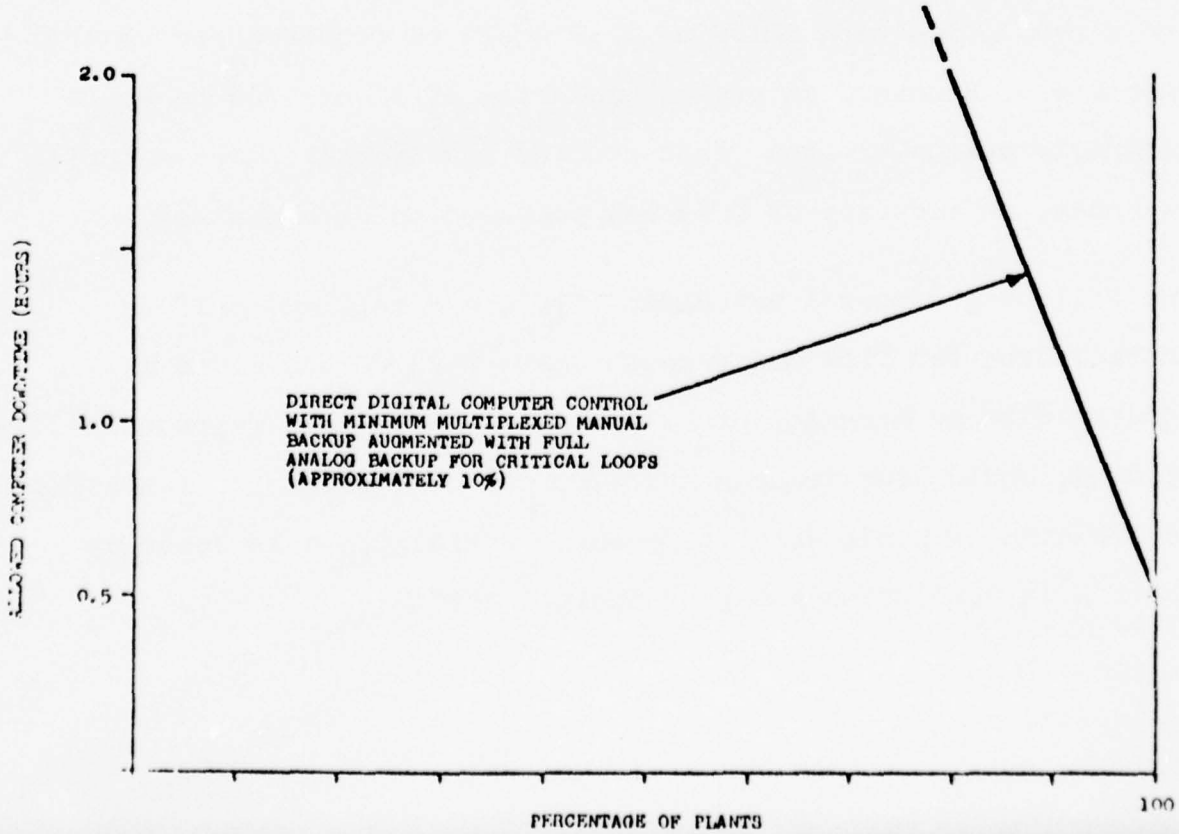
   (2) Control response information - variables sampled each several seconds, allow recall for from one-half hour to one hour. (Will comprise about 25-30% of loops - should be selectable by dialing into the program.)

A third topic discussed was the desirability of using conventional analog controllers for critical loops as part of the backup system for the computer to relieve the burden on the operator during the manual operation period. It was felt that 5-10% of such permanent backup loops might be desirable for some especially sensitive operations. The effect of using such controllers upon the allowed downtime for computer repair is shown on Figure 8.

FIGURE 14

RELATIONSHIP OF ALLOWED COMPUTER DOWNTIME PER FAILURE TO TYPE OF BACKUP USED





QUESTION 16 (Previous QUESTION 36) -- What are the desirable specifications for accuracy, repeatability, and reliability for process control components other than computers? Would we pay a premium over current prices to obtain such characteristics?

ANSWER -- The Guidelines calls for an overall computer input resolution and computational accuracy of 0.1% of full scale reading. It is generally agreed that present day commonly available control computers have about a 1% of full scale accuracy for a typical loop configuration when properly calibrated. Under normal operational conditions this rapidly deteriorates to 2-5% of full scale. Repeatability and resolution are better than this-being of the order of 0.5%.

The workshop attendees agreed that eventually a reproducibility and a resolution capability of 0.1% would be required for control loop uses. However, an overall accuracy of 1% was judged to be generally adequate. For yield studies and overall plant material balances, an accuracy of 0.1% was conceded to be necessary.

The following proposal was made: "If a d/p cell and orifice installation for flow measurement costs $375.00 and gives a reading with an accuracy of 2% and a resolution and reproducibility of 0.5%, would your company pay $700.00 for a flowmeter installation of the same or a different type which would give a 1% accuracy and a 0.1% resolution and reproducibility?"

1) For 50% of the flow loops?

Yes: 9 No: 9

2) For 25% of the flow loops?

Yes: 14 No: 4

SECTION VI

STATE-OF-THE-ART OF PROCEDURAL PROGRAMMING LANGUAGES FOR INDUSTRIAL APPLICATIONS

Along with the review of the state-of-the-art of data collection, computerized system check-out and sequencing control functions given in Section III of this Part, the attendees at the First Meeting of the Purdue Workshop on Standardization of Industrial Computer Languages developed the attached report on the state-of-the-art of procedural programming languages for industrial applications. This report was published as pp. 25-52 of the Minutes of that Meeting.

STATE-OF-THE-ART OF PROCEDURAL PROGRAMMING LANGUAGES FOR INDUSTRIAL APPLICATIONS

INTRODUCTION

The Procedural Programming Language Committee of the Workshop on Standardized Industrial Control Programming Systems was formed to investigate the possibility of standardizing procedural programming languages for industrial control.

The major thrust of the effort developed along two lines: development of some feeling for the desirable features in such a language, and development of a feeling for the standardization process.

The feeling evolved that our real problem is not in the lack of a standard, but in the lack of uniformity or compatibility of programming systems. Standardization is a long and arduous process which would ultimately solve this problem, but more immediate steps are also necessary to help ease our present problems. It is to both these concepts that this report addresses itself.

The report is in three parts, with two appendices. Part One summarizes existing or proposed features in process control programming languages (or real time languages) which are deemed by the Committee to be of interest after review. Part Two examines existing languages in this framework, identifying areas in which compatibility of these languages could be achieved.

PRECEDING PAGE BLANK-NOT FILMED

Part Three proposes an approach to the achievement of a process control language which would be worthy of consideration for a standardization. The appendices include: A summary of the written proposals received before the meeting, and a discussion of general user requirements (any proposed language must be measured ultimately in terms of the manner in which it meets such requirements).

1. SUMMARY OF THE FEATURES OF THE PROCEDURAL LANGUAGES PRESENTED.

This section is a summary of the salient features of the procedure oriented languages discussed at the orkshop. The languages covered by the vendors were: FORTRAN (BAILEY, CDC, FOXBORO, GE, IBM, L&N, SEL, WESTINGHOUSE), AUTRAN (CDC), SPL-IV (L&N), INDAC, a modified BASIC, (DEC), and REAL TIME FORTRAN (HONEYWELL). The other languages discussed included SPL/JOVIAL 6, RTL (British) and RTL (Case Institute).

A few brief comments are included below concerning the summary. A majority of the languages are based on FORTRAN or FORTRAN like languages. Few of the languages exactly meet the full USASI FORTRAN specifications. All of the FORTRAN based languages presented appear to have implemented a large subset of the full USASI FORTRAN. A lack of uniformity exists in the various implementations of data structures, data manipulation and file handling. However, considerable uniformity exists in the functions for program control.

It is noteworthy that there was little emphasis in the discussions on the handling of I/O, and the concept of concurrent I/O and compute capability within the same program.

The features are described under the following headings:

1. Data Structures and Variables
2. Input-Output Features
3. Compiler Operation
4. Program Control
5. Special Features

(Not all of the features discussed below have been implemented).

1.1 Data Structures and Variables

1.1.1 DATA TYPES

.1 Fixed Point Data. Allows arithmetic to be done with scaled, mixed numbers. (Such as a decimal number 123.45 scaled by 2^8). The compiler keeps track of the binary point during arithmetic operations and shifts the variables accordingly while using integer arithmetic on the variables. No example was shown of how this was actually implemented.

.2 Bit, Status, and Byte Variables. A class of variables having a fixed range of values which can be represented in fewer bits than one computer word. This class occurs frequently in process control. Many variables may be packed in one word to improve efficiency. Techniques to specify the assignment of these variables to bit positions within a word were presented (BYTE statement).

.3 Number Bases. The capability of specifying numeric information in several different number bases was presented. Binary (2), Octal (8), Decimal (10), and Hexidecimal (16) were all used.

1.1.2 DATA MANIPULATION

.1 Byte and Bit Manipulation. These are implemented by 1) subroutine calls, such as CALL SHIFT(var,count); 2) additional statements RESET BIT(var,bit pos.); 3) arithmetic replacement statement I=byte variable; and 4) logical operators such as OR, AND, EOR, NOT

.2 Character Manipulation Techniques. These are implemented through subroutines calls as in CALL ENCODE(params).

.3 String Manipulation. Techniques for manipulating strings of data were presented in RTL (copy statement).

.4 Stack Manipulation. Techniques for manipulating stacks of data were presented in RTL (circular list concepts).

1.1.3 VARIABLE NAMES

Variable names of length greater than 6 characters were used in the following languages: AUTRAN (16), SPL-IV (8), DEC (unlimited), GE (8), Westinghouse (8), Bailey (8). Almost all of the FORTRAN based languages permit up to 6 character variable names. Some of the languages used coded names to specify process variables and thus become reserved words, such as the name AI001 meaning analog input point 1. This feature is used to eliminate repetitive type declarations.

1.1.4 ARRAYS

No proposal was discussed to allow more than three dimensions to an array.

1.1.5 DATA AREA SPECIFICATION

The global COMMON statement was used to permit data to be used by anyone in the system. This concept was implemented in most FORTRAN based languages, however, it was not used uniformly. The statements:

```
GLOBAL COMMON A
COMMON/INSKEL/A
COMMON  A
```

have the same effect on different systems.

1.1.6 FILES ON BULK MEMORY

Several languages provided file generation, file handling and Read/write of bulk memory. Some systems allowed file generation (allocation) only at system generation time. The method of accessing the files was implemented in the following ways:

1. subroutine calls to a file handler as in CALL GETFIL(parameters)
2. statements added to the languages, as OPEN, DEFINE, etc.
3. modification to existing statements, READ(file name(record no), format) list and READ DISK.

Both sequential access file and random access files were described. A technique of packing the information in the file was also described.

It should be noted that there is little uniformity in the way this function was implemented in the various languages. No one discussed formatted I/O on bulk memory even though it exists in several FORTRANS.

1.2 Input-Output Features

.1 READ/WRITE to bulk storage. This topic is described under file handling.

.2 READ/WRITE to non-process I/O equipment. This capability appeared to be available in most of the languages presented.

.3 READ/WRITE of process I/O equipment. This capability was implemented in the FORTRAN based languages through subroutine calls or statements such as INPUT or OUTPUT.

.4 READ/WRITE of other computers. This capability was not described.

.5 Specification of I/O devices in the language was done in AUTRAN and in FORTRAN with statements like DEVICE... and EQUIP...

1.3 Compiler Operation

.1 SUFFIXES. Suffixes to variable names were used to effect data transformations in only one language (SPL-IV). The suffixes behave somewhat like arithmetic operators on the base variable so that AIOO1AV is an averaged value of AIOO1. Average, accumulate, average over 2 minutes, and other suffixes were implemented.

.2 Decision Tables are included in only one language (SPL-IV).

.3 Conditional Compilation. TRACE/UNTRACE. Additional statements to provide debugging tools were introduced in some of the languages.

.4 Compile Time Directives. The RELATIVE statement a variation on the EXTERNAL statement permits the compiler to generate relative address references to the specified routines. This allows dynamic storage allocation techniques to be used without modifying the object code at execute time.

.5 Reserved Words. The subject of reserved words was discussed only in SPL/J6.

1.4 Program Control

The requirement for the programmer to control the sequence of program execution was implemented with a high degree of uniformity in most of the FORTRAN based languages.

.1 Scheduling of Programs. This was handled by calling on a scheduling function. It was implemented by either a subroutine call or by a statement added to the language. Most of the systems provide for the stacking or queueing of the requests for execution of other programs.

.2 Linking Interrupts to Programs. This was implemented by either a subroutine call or by a statement such as CONNECT OR DISCONNECT.

.3 Time Delays. These were implemented by subroutine calls or by a statement such as WAIT.

.4 Clock. Access to the clock was provided in most systems.

.5 Program Termination. This was accomplished by either a subroutine call or through a statement such as STOP, or CALL EXIT.

.6 Parallel Task Execution. This was provided in most languages. Temporary suspension of a program was accomplished by either subroutine calls or by an additional statement. An extended form of the READ/WRITE statement was also used to permit parallel task execution as in READ(lu,format) list, label, priority. Also exists in PL1.

.7 Multiple Subroutine Returns. This topic was not discussed.

1.5 Special Features

.1 I/O Error Recovery. The READ/WRITE statement was modified to include the return of error information as in READ (lu,format,error) list.

Other implementations allow the user access to the status of the last I/O request through an integer function as in IERR = IOCHK(0).

2. RECOMMENDED ACTION TO INCREASE COMMONALITY IN EXISTING PROCEDURAL LANGUAGE

From the proceedings of the Workshop it is evident that there are a number of FORTRAN based Industrial Programming Languages. Although these languages are based on a common subset, their real-time extensions are varied.

For example, in the five areas identified in Table 2.1 there are common functions represented by different formats.

It is the expressed desire of a number of the attendees at the Workshop that an effort be made to extend the compatability of these languages.

Therefore, the Procedural Programming Language Committee recommends the formation of a committee whose goal it will be to achieve further compatability in existing FORTRANs. To that end we suggest that the following three functions be performed by the committee.

1) The feasibility of achieving uniformity of the functions of Table 2.1 will be investigated, and preliminary steps for doing this where possible taken.

2) The committee will determine the exact extent of the compatability of these existing languages.

3) The committee will propose areas in which the compatability of these languages may be extended, will attempt to define a machinery for achieving this, and will set this machinery in motion, if appropriate.

TABLE 2.1

COMPARISON OF FORTRAN EXTENSIONS

VENDOR	TIMER HOLDS CONTINUATION OF A PROGRAM FOR A SPECIFIED LENGTH OF TIME	SCHEDULER INITIATES ANOTHER PROGRAM AT A SPECIFIED PRIORITY
BAILEY	DELAY	ACTIVATE
CDC	ASSIGN 100 TØ ICØMP CALL TIMER (ICOMP, units, level + selection of interval, 4 word temp. block) CALL DISPAT 100 CONTINUE	CALL SCHEDL (pgm name, parameter, level, 4 word temp. block)
DEC	ACTION #1 Do Snap #4 every 1 SEC TIMER (START, #1) STOP	EXIT then do Snap #4
GE	DELAY PROGRAM, m_1, m_2 where m_1=1 if this program's area of occupancy is available to another program =0 if area is to remain unavailable m_2=length of delay	TURN PROGRAM n ØN, expression where expression = next time of initiation, with zero indicating immediate execution
HONEYWELL	DELAY EXECUTION It is implemented via CONNECT TIMER and SCHEDULE statement	SCHEDULING ACTIONS REQUEST PROGRAM (parallel task execut) REQUEST PROGRAM or REQUEST PROGRAM ($a_1,\ldots,a_n$) or REQUEST PROGRAM ($a_1,a_2,\ldots a$)100 non-sequent. or REQUEST PROGRAM ($a_1,\ldots,a_n$)100,2 softer level Schedule label (parallel path in a pgm) SCHEDULE 100 Statement #
IBM	CALL SUSPN CALL DEFEN CALL DELAY CALL REPET CALL CYCLE CALL CANCL	CALL LINK CALL SPECL (BACK) CALL QLEVL CALL QUEUE (TSX) CALL LEVEL CALL ISPAR
WESTINGHOUSE	CALL M:STIMER ( ) S:DELAY ( )	CALL M:TASK ( )

TABLE 2.1, cont.

PROGRAM TERMINATION TERMINATES PROGRAM EXECUTION AND RELEASES CORE WHERE APPLICABLE	PROGRAM SUSPEND HOLDS CONTINUATION OF PROGRAM UNTIL AN EXTERNAL EVENT OCCURS (e.g., I/O)	CONNECT PROGRAM TO INTERRUPT ASSOCIATES A PROGRAM TO A SPECIFIED INTERRUPT
STOP CALL RELEASE (pgm. name)	SUSPEND CALL DISPAT STØP	CONNECT ASSIGN 100 TØ IØC CALL SETLINE (line, level, IØC)
EXIT	Implicit in ACTION section PAUSE (?)	Implicit in SET and SEND statements PHASE (lu_1,lu_2,...,lu_n) GET (INT) IA GO TO (k_1,k_2,...,k_n), IA
TURN PROGRAM OFF m_1,m_2,m_3, RETURN m_4 where m_1,...,m_4 = 1 or 0 and specifies whether certain statuses will exist at next initiation of program RETURN = location for program entry at next initiation	No specific single statement Can use a DELAY and TEST	None at Fortran level
TEST PENDING or TEST PENDING 100	TEST PENDING or TEST PENDING 100	INTERRUPT JOE CONNECT INTERRUPT JOE (a) CONNECT TIMER JOE (a_1,a_2,a_3) CONNECT CLOCK JOE (a_1,a_2,a_3,a_4)
CALL EXIT CALL LINK CALL CANCL	CALL SPECL (CALL BACK) PAUSE CALL SUSPN	(Connected in a system generation) or (LINK-EDIT-PHASE) CALL MASK CALL UNMSK
CALL EXIT	Not Applicable-Suspension is not subject to program control, but is handled automatically by the operating system.	Not Applicable-This is accomplished at system generation and handled automatically by the operating system.

3. RECOMMENDATIONS OF THE PROCEDURAL LANGUAGES COMMITTEE REGARDING FURTHER LONG TERM DEVELOPMENT WORK

3.1 The capabilities desired for procedural languages can be categorized a short-term, intermediate, and long-term. There are several functions, such as FORTRAN calls to scheduling routines, which appear in various forms in existing real-time FORTRANS. Achieving uniformity for these functions is a short-term goal. In the intermediate time, there are possible FORTRAN extensions, such as bit manipulation and byte equivalence for which uniformity can be attained through subsequent work of standards committees. Many of the Workshop attendees are convinced that the requirements for a standard procedural language for industrial computers can be satisfied by this approach.

3.2 But many attendees feel that the question of standard languages for industrial computers should not be restricted to FORTRAN extensions, although these should be functionally preserved. This is not in conflict with the short-term and intermediate goals. PL/1, variants of RTL, and the NASA Space Programming Language have been discussed as examples.

3.3 In order to conform to the adopted goals of the Workshop, with specific reference to the development and use of a standard high-level language, a committee is to be formed, and is directed to take the following action:

1. Consolidate the language features (Table 3.1) into objectives and criteria for a procedural language.
2. Examine the relationship of these features to PL/1, RTL (and others), because of the apparent similarity of the specifications to these languages.
3. Develop a plan for:
   i) Specification of the language
   ii) Trial implementation
   iii) Certification of this language as an official standard

3.4 As a guideline for such a committee, the following summarizes the types of features to be considered, as expressed during the Workshop:

.1 General:

Cost and ease of implementation

Natural language and syntax for industrial users

Consider commercial security, maintenance of software systems and compilers

.2 Job Management:

Interrupt Response Procedures

Real-time Facilities

Ability to Call Subprograms in Other Languages

Error Detection and Recovery Procedures

Process Task Supervision

Off-line Testing

Memory Allocation

.3 Data Management:

Data Types and Conversion

Logical
Fixed
Status

Decision Tables

Data Structures (Table Packing)

Ability to Access, Maintain, and Manipulate Files

Compatability with Management Information Systems

Machine to Machine Data Interchange

On-line Parameter Manipulation

.4 Compiler Features:

On-line Compilation

Upward Compatability Over the Range of Applicable Machines

Program Simulation on Other Machines

Estimation of Run-times

Compile-time Computation

Editing and Documentation Aids

Cross-references:

between statements and variables

for process variable interactions

Source Language Diagnostics

.5 Algorithmic Capabilities:

Ability to Express and Execute Arithmetic and Logical Operations Easily

Manipulation of Symbol and Logic Strings

Logging and Reporting

Decision Table Operations

.6 Input/Output Functions:

Data Acquisition

Queueing and Priority Assignment of I/O Resources

Process Output Communication

Console and CRT Communication

REPORT APPENDIX I

SUMMARY OF USERS COMMENTS

REPORT OF SUBCOMMITTEE OF THE COMMITTEE ON PROCEDURAL LANGUAGE

This report is a summary of the comments on standardized software submitted to Professor T. J. Williams in response to his letter of January 24, 1969. Seventeen companies submitted comments, fourteen users, two universities and one vendor. The comments were essentially of two types; specific requirements usually reflecting known deficiencies of existing systems; and user functional requirements with no specific statement of how these goals are to be met. We have divided our summary into two classes called requirements and goals. These are listed below with the number in parenthesis giving the number of companies who referred to this point. The comments summarized below are included along with several additional ones received during the Workshop as Minutes Appendix II.

A. ROOT LANGUAGE

1. The root language should be FORTRAN (10)
2. The root language should be RTL (1)
3. The root language should be flowchart-oriented (1)

B. DATA STRUCTURES AND VARIABLES

Data Types:

1. Bit and byte declaration. (8)

2. The ability to express a data quality field within a real variable. (2)

3. A literal representation of a byte as octal, hex or equivalent. (2)

Variables:

4. Variable length names (1)

5. Standard labeling of variables representing process points (with a limit of 12 characters) (3)

Arrays:

6. N-dimensional matrix manipulation (1)

7. Variable length buffers (1)

Data Areas:

8. Ability to specify data areas as resident or non-resident (1)

9. Global and labeled common perpetually identifiable by name (1)

Files:

10. File handling (2)

11. File organization for process I/O compatable with EDP I/O (1)

Special Data or Declaration:

12. A system generation procedure to tailor to a particular configuration (at load time?) (at compilation time?) (1)

C. INPUT-OUTPUT FEATURES

1. All I/O devices, in all their modes, should be under program control through standard I/O statements. (7)

2. Parallel I/O processing of all I/O devices (1)

3. Standard input format (1)

4. Control action of certain devices should be specified; e.g., width of page, etc. (1)

5. Specialized format statement for logging and alarming (1)

6. I/O initialization after compiling (1)

D. COMPILER OPERATION

1. In-line assembly coding permitted (4)

2. Consistent compiler action on encountering ambiguities (1)

3. Absolute object coding allowed (1)

4. Incremental additions without recompiling (1)

5. Conditional compilation for debugging purposes (1)

6. Program specification or control of program overlays at compilation (1)

7. Simulator for use with on-line data (1)

8. Symbolic coding at subroutine level only (1)

E. PROGRAM CONTROL

1. Timers and clocks (3)

2. All hardware interrupts accessible and controllable by programming including such functions as service, inhibit, generate, permit (3)

3. Program segmentation inherent in the language (1)

4. Every program to be protected from the errors of other programs (1)

5. Re-entrancy and program precision variable within a program (1)

6. Adequate communication with the executive (1)

F. SPECIAL FEATURES

1. Ability to handle standard conversions such as thermocouples, linear signals, square root, orifice plate, etc. (2)

2. Ability to specify and handle multi-processor environment (2)

3. Decision tables (2)

4. Sequencing capability (STEP,CHECKPOINT, etc.) (1)

GOALS

Noted from the pre-session write-ups were the following goals reflecting desirable attributes of the subject procedural language. In addition a by-product capability is suggested which would enable Executive System Generation to be performed either within the subject procedural language or by using another procedural language.

1. Fast-efficient execution of the generated object code was called for.

2. Utilization of all hardware features of the object computer was suggested, e.g., index registers, move instructions, etc.

3. Efficient use of object computers memory (both main and auxiliary) by the object code was suggested.

4. Included in the subroutine library should be process and statistical functions, e.g., operators console and report generation.

5. Object code operable as DDC algorithms should be capable of being produced using the procedural language.

6. Communication between computers, including time-shared computers, should be facilitated.

7. A structure permitting subsets of the language to be available for limited machine configurations is a design criterion of the language.

8. Other special purpose compilers should have the capability of calling on the subject procedural language to handle operations outside their scope.

9. The language should handle the general control problems of an integrated plant but should not be process dependent.

10. Easy programming should be possible.

11. The language should be a new development and not an extension or superset of FORTRAN.

12. Inclusive in the scope of the language should be the problems of one-of-a-kind research installations.

13. Output of the compiler should include indications of object program size and execution time. In addition meaningful diagnostics involving process diagnostics should be included.

14. The language should use a META compiler philosophy.

15. A real-time executive should be able to be written using the subject language (this is not the suggestion that some language be provided to enable system generation).

REPORT APPENDIX II

SUBCOMMITTEE REPORT SUMMARY OF APPLICATION REQUIREMENTS

This report summarizes the information contained in 19 Application Questionnaires completed by industrial system users. The purpose of the report is to provide some measure of (a) the distribution of industrial systems in the various industries and applications, and (b) characteristics of existing systems. Basis for comparison of the relative frequency of subcategories is provided as simple percentages of total responses within each category, rather than weighted averages (e.g., by the number of systems owned). A copy of the original Questionnaire is attached.

1. Type of industry

1.1 Questionnaires filed by:

Chemical & Petroleum	50%
Other (1)	50
	100%

1.2 Systems used by

Chemical & Petroleum	53%
Other (1)	47
	100%

Note: (1) Other users include Metal, Paper, Glass, Discrete Manufacturing, Power, etc., industries.

2. Type of process

2.1	Continuous	30%
2.2	Non-continuous	27
2.3	Both (start-up, shutdown)	28
2.4	Other	15
		100%

3. Type of control

3.1	Open loop	26%
3.2	Closed loop	44
3.3	Data Acquisition and/or Monitoring	25
3.4	Optimizing	5
		100%

4. Control system characteristics

4.1 Distribution of DDC systems:

Number of Loops	
up to 100	78%
101-300	11
over 300	11
	100%

4.2 Input/output distributions by %.

	Analog		Digital	
	Input	Output	Input	Output
None	24	46	24	24
Less than 129	31	43	33	45
129-256	17	7	10	4
257-512	10	4	3	10
513-1024	10	0	17	7
Over 1024	8	0	13	10
TOTAL	100	100	100	100

	Event Counters	External Interrupts	Data Links
None	61	47	81
Less than 17	36	33	19
17 or over	3	20	0
TOTAL	100	100	100

4.3 Distribution of allowable response time to external stimulus.

4.3.1 Digital signals, including interrupts

Less than 1ms	1%
1-10 ms	2
10-100ms	40
100-1000ms	30
1-10 sec	27
TOTAL	100%

4.3.2 Analog signals

less than 1 sec	6%
1-10 sec	6
10-60 sec	71
1-5 min	16
Over 5 min	1
TOTAL	100%

5. System configuration

5.1 Core size distribution

Core size (1000 words)	
Less than 8	25%
8-16	21
16-24	10
24-32	44
TOTAL	100%

5.2 Mass memory availability

Not available	31%
Drum or Disk	69
TOTAL	100%

5.3 Mass memory size distribution

5.3.1 Drums

Size (1000words)	
Less than 128	54%
128-256	23
256-512	23
TOTAL	100%

5.3.2 Disks

Size (1000 words)	
Less than 500	32%
500-1000	27
1000-1500	27
over 1500	14
	100%

5.4 Peripheral equipment distribution

5.4.1 Typers

Count	
None	3%
Less than 5	66
5 or over	31
TOTAL	100%

5.4.2 Operator's Console

Count	
None	23%
One	44
Over one	33
TOTAL	100%

5.4.3 CRT-s

Count	
None	72%
Over 3	28
TOTAL	100%

6. Software

6.1 Languages used in existing systems

Assembler	44%
FORTRAN	39
Other	17
TOTAL	100%

6.2 Routinely used background compilation

Yes	57%
No	43%
TOTAL	100%

Note: 86% of the systems had background foreground separation available.

6.3 Source of software

Source	
Vendor, as is	49%
Vendor, modified	16
Home made	35
TOTAL	100%

7. General

7.1 System cost distribution

Cost ($000's)

Less than 100	35%
100-500	42
Over 500	23
TOTAL	100%

APPLICATIONS QUESTIONNAIRE

The Committee on Procedural Languages solicits your help in determining application requirements for Industrial Computer Programming Systems.

Please complete the attached questionnaire and return to T. G. Gaspar - including your name and affiliation. If we missed a point you consider important please add your comments.

Name_____

Affiliation_____

1. Type of industry

Chemical & Petroleum	____	
Metal	____	
Paper	____	
Other	____	Type: _____

2. Type of process

Continuous	____	
Non-continuous (batch, etc.)	____	
Both (start-up, shutdown)	____	
Other	____	Type: _____

3. Type of control

Open loop	____	
Closed loop	____	
Data Acquisiton and/ or monitoring only	____	
Other	____	Type: _____

4. Control System Characteristics of Typical System

4.1 Input/Output count

	Existing		Planned	
	min	max	min	max
Analog Inputs	____	____	____	____
Analog Outputs	____	____	____	____
Digital Inputs	____	____	____	____
Digital Outputs	____	____	____	____
Event Counters	____	____	____	____
External Interrupts	____	____	____	____
External data links (telephone lines to MIS, etc.)	____	____	____	____

4.2 Maximum allowable (a) response time to external stimulus, or (b) scan rate

	Number of Points			
	Analog Signal		Digital Signal (including interrupts	
	min	max	min	max
less than 1 ms	____	____	____	____
1-10ms	____	____	____	____
10-100ms	____	____	____	____
1-10 sec	____	____	____	____
10sec-1min	____	____	____	____
1-5 min	____	____	____	____
greater than 5 min	____	____	____	____

4.3 Existing System Configuration(s)

Single computer _____
Dual computer _____

	min	max	Essential for operation*
Core Size, words	___	___	_____
Drum Size	___	___	_____
Disk Size	___	___	_____
Number of Typers	___	___	_____
Printers	___	___	_____
CRT's	___	___	_____
Operator's Console	___	___	_____

*A device is essential if its failure causes system failure. Please indicate if you use other devices such as line printers, card I/O, etc.

4.4 Software in existing systems

4.4.1 Languages

ASSEMBLER _____
FORTRAN _____
Other _____ (specify other_____)
ROUTINELY USED BACKGROUND COMPILATION (yes)___ (no)___

4.4.2 Realtime Software

	Supplied by vendor	Modified	Homemade	None
Operating System				
Background-Foreground Separation	_____	___	___	___
Analog Scan	_____	___	___	___
Digital Scan	_____	___	___	___
DDC & Cascade	_____	___	___	___
(No. of DDC loops:____)	_____	___	___	___
Supervisory control				
Other (specify other_____)	_____	___	___	___
	_____	___	___	___

5. General

Approximate cost of (Smallest existing system $_____
Largest existing system $_____

Number of systems_____

If demonstrations of software were available at no great financial burden for the vendors, would you be interested?

Yes:_____ NO:_____ Do not care:_____

6. Comments

SECTION VII

SOME ADDITIONAL IMPORTANT TOPICS PRESENTED TO THE WORKSHOP REGARDING THE STATE-OF-THE-ART OF APPLICATIONS

The attached three reports are collected together because of their relatively small size. Their content is however very important in reviewing the state-of-the-art of the industrial computer applications field.

The first report is a new document not yet published in a Workshop Minutes. It reports a survey recently conducted in Japan to determine industrial interests in various reliability techniques.

The second report, also new, is a survey of present and projected process computer applications and research activities in West Germany.

The third document from pp. 105-111 of the Minutes of the Second Purdue University Meeting of the ISA Computer Control Workshop is an interesting compilation by Mr. R. L. Curtis of the pertinent time constants and "dynamics" of various processes and various management functions.

REPORT ON JAPANESE INDUSTRIAL COMPUTER SYSTEM QUESTIONNAIRE SURVEY

GENERAL

This survey was carried out to investigate the reliability and safety of industrial computer sustems now operating in Japan, and also to ascertain what levels of reliability and safety users will require of such systems in the future.

The questionnaires were sent to Japanese users and manufacturers of industrial computer systems, and 149 questionnaires were returned.

The questions referred to a wide scope of problems concerned with industrial computer system safety, in order to obtain fundamental data for the planned activities of our committee (JEIDA's Sub committee for Safety and Security of Industrial Computer Systems).

The questions are broadly divisible into the following categories.

(1) Background Information
(2) System Configuration
(3) Reliability
(4) Installation Evironmental Conditions
(5) Operator Responsibilities
(6) Safety Assurance
(7) Maintenance
(8) Cost/Safety Trade-Offs

The most significant of the survey findings are presented in this paper. Although there were many responses from manufacturers, this paper stresses the information obtained from user responses.

PRECEDING PAGE BLANK-NOT FILMED

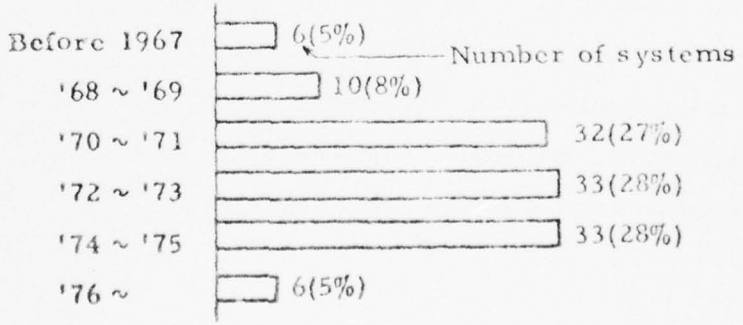
RESULTS OF SURVEY

1. Background Information

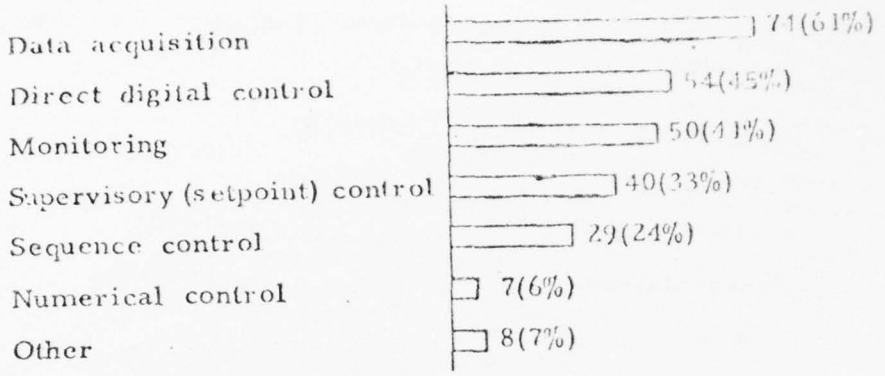
1.1 The number of repliers and their types of industry

Industrial Classification		Number of companies	Number of systems surveyed
Users	Petroleum refining	8	14
	Petrochemical	29	33
	Steel	12	17
	Nonferrous metals	8	8
	Textile	4	5
	Water supply and sewage treatment	5	6
	Food	2	2
	Gas and Power	13	15
	Rubber, Pulp and paper	6	6
	Transportation	7	10
	Other	5	5
	Total	99	121
Manufacturers		19	28

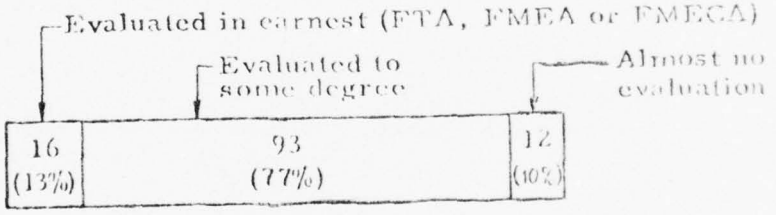
1.2 When was your system installed?



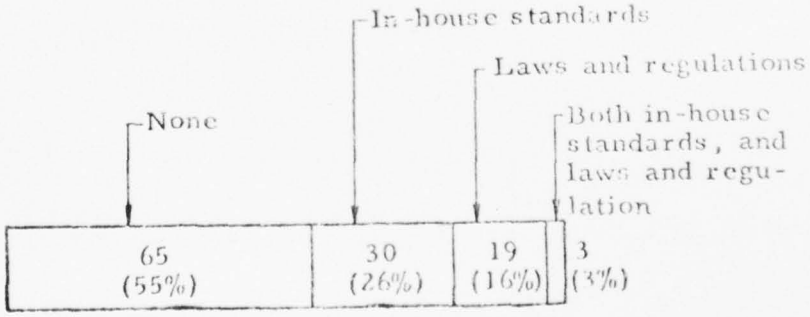
1.3 What is the major function of your system?



1.4 Did you evaluate the safety of your system, when you planned to install it?



1.5 What sorts of laws and/or standards did you refer to, in planning, designing, and installing your system?



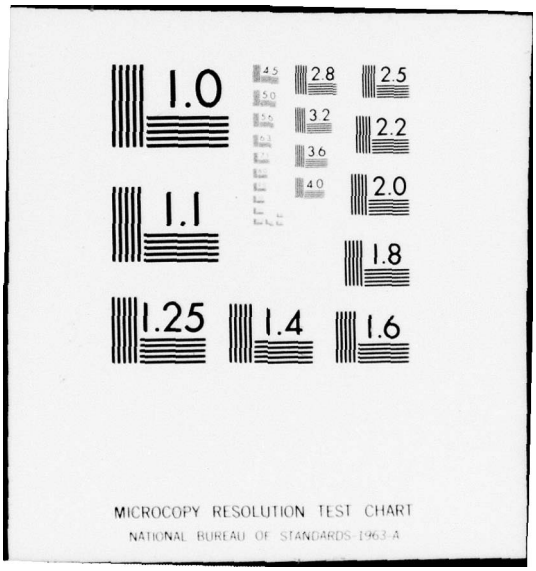
AD-A038 058 PURDUE UNIV LAFAYETTE IND PURDUE LAB FOR APPLIED IND--ETC F/G 9/2
SIGNIFICANT ACCOMPLISHMENTS AND DOCUMENTATION OF THE INTERNATIO--ETC(U)
JAN 77 N00014-76-C-0732

UNCLASSIFIED NL

4 OF 4
ADA038058

END

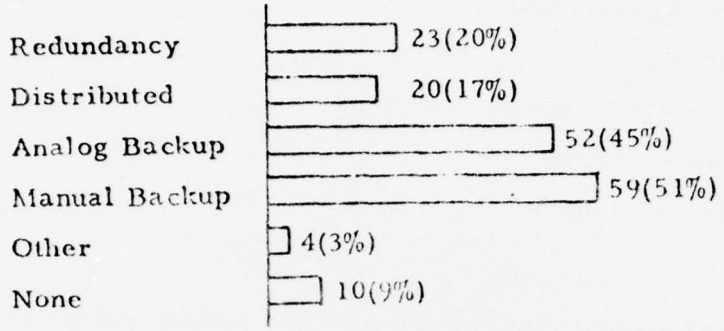
DATE FILMED
5 - 77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

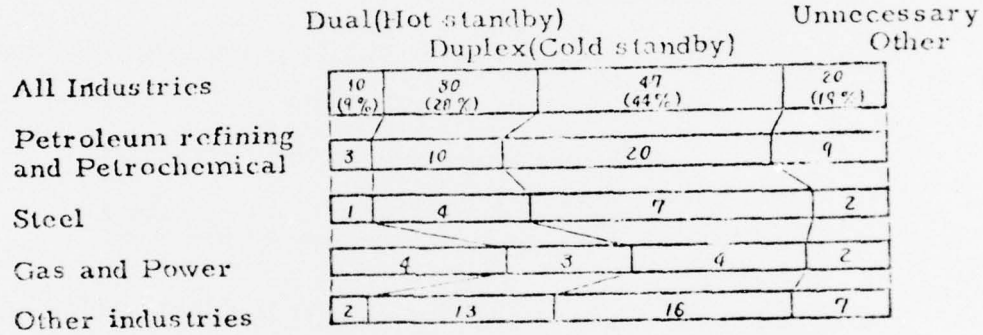
BEST AVAILABLE COPY

2. System Configuration

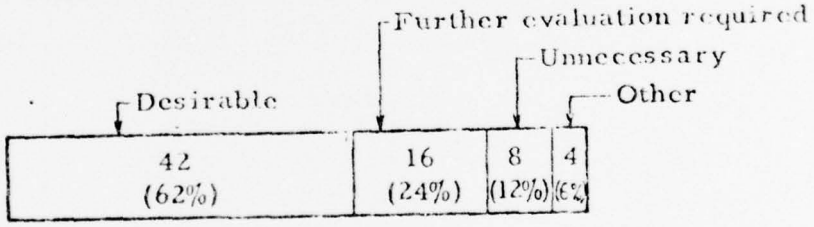
2.1 What sort of techniques did you adopt in your system in order to increase reliability and safety?



2.2 What kind of redundant system will you require in the future?



2.3 What do you think about distributed systems, in terms of system safety?

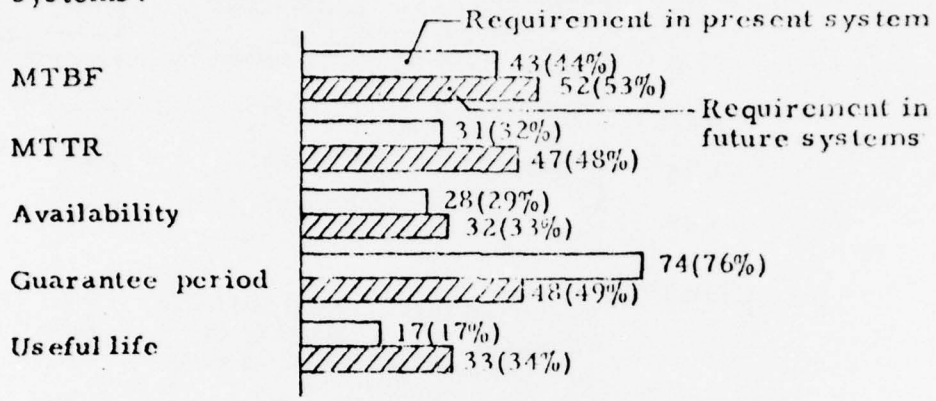


The majority of users regard distributed systems as contributing to safety, but they point out problems such as cost, complexity of system configuration, reliability of the communication lines among stations, etc.

BEST AVAILABLE COPY

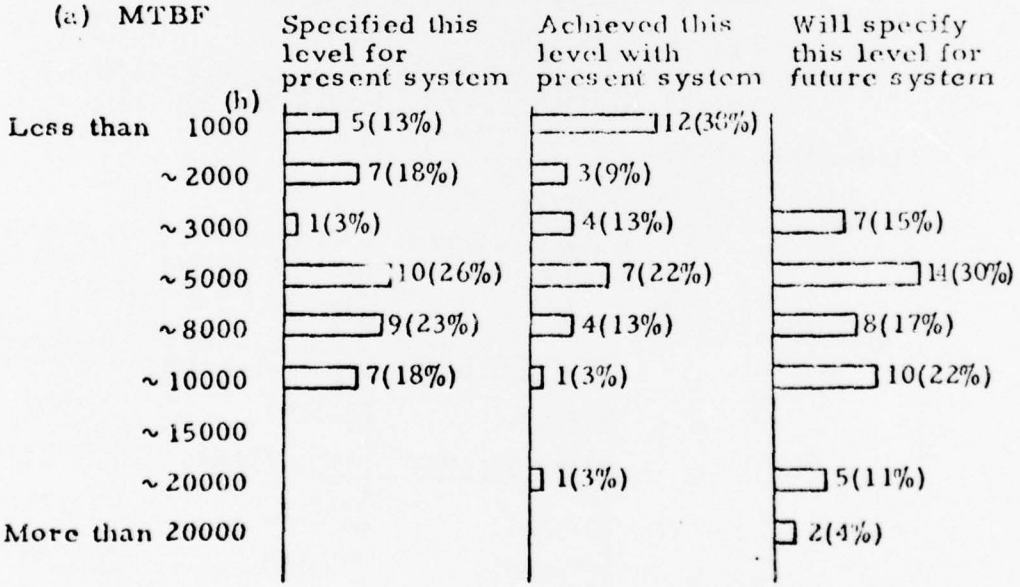
3. Reliability

3.1 When acquiring your present system, did you require the manufacturer to meet (or furnish proof of his ability to meet) MTBF, MTTR, availability, and useful life goals? Did you require a guarantee period? What will you require for future systems?

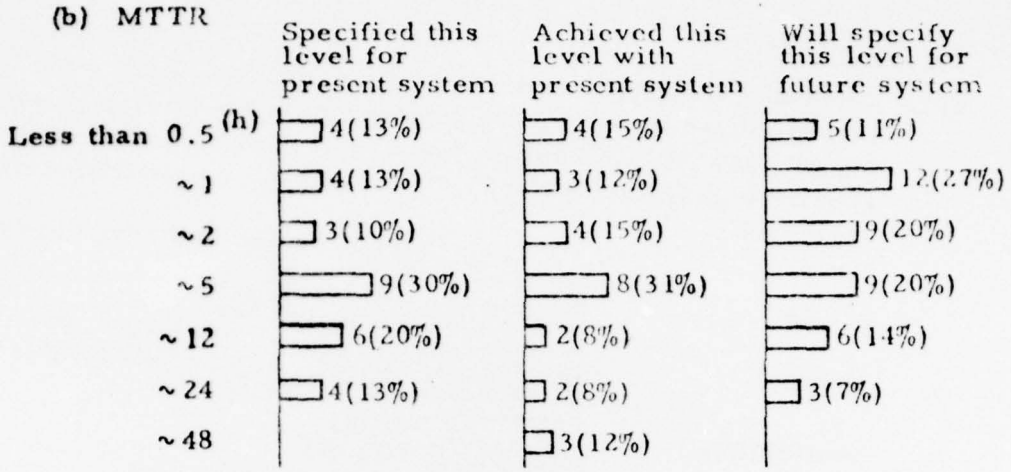


3.2 What numerical values did you specify for the requirements of the above. What were the actual results?
What values are you going to specify in the future?

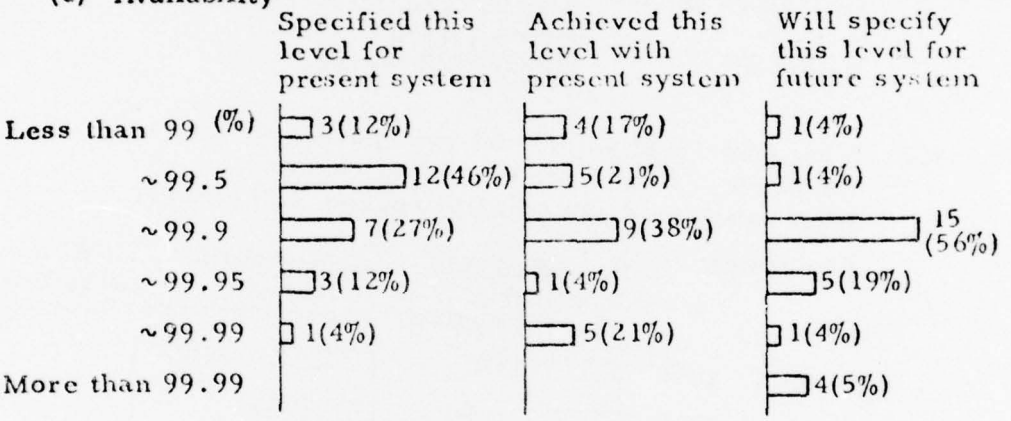
(a) MTBF



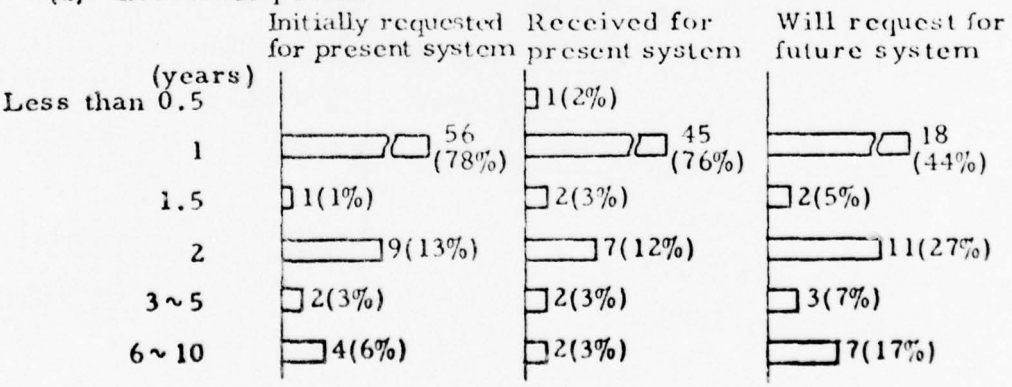
(b) MTTR



(c) Availability

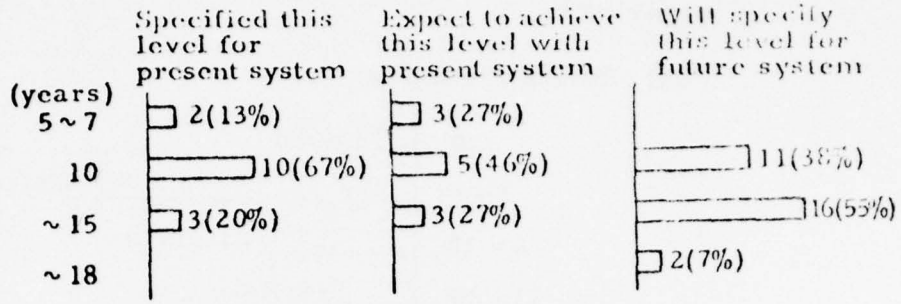


(d) Guarantee period

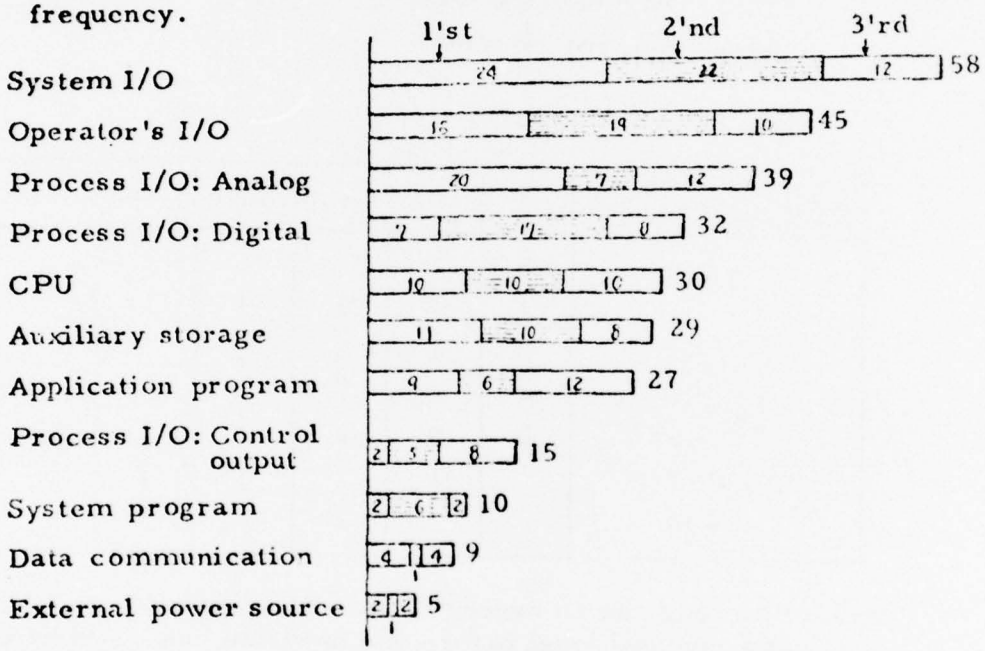


BEST AVAILABLE COPY

(c) Useful life

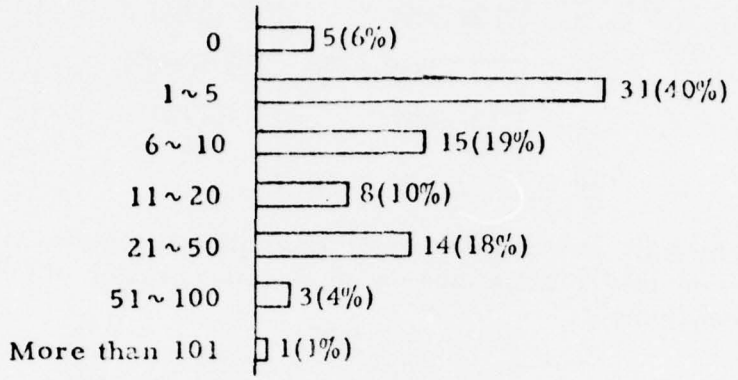


3.3 Which three computer system components have failed most frequently? Rank these components in order of failure frequency.



3.4 How many program bugs have you found since actual operation started?

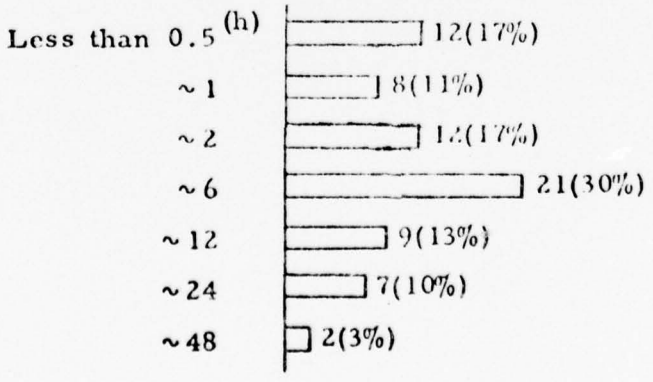
(a) During First Year



(b) After first year

Length of period (years) \ Number of bugs	0	1~5	6~10	11~20	21~50	51~100	More than 101
0.5	4	1					
~1	1	7	6	1			
~2	2	6	3				
~3	4	2	5	2	1		1
~5	2	9	4	3		1	
~7		1	1	1	2		
~10			2				

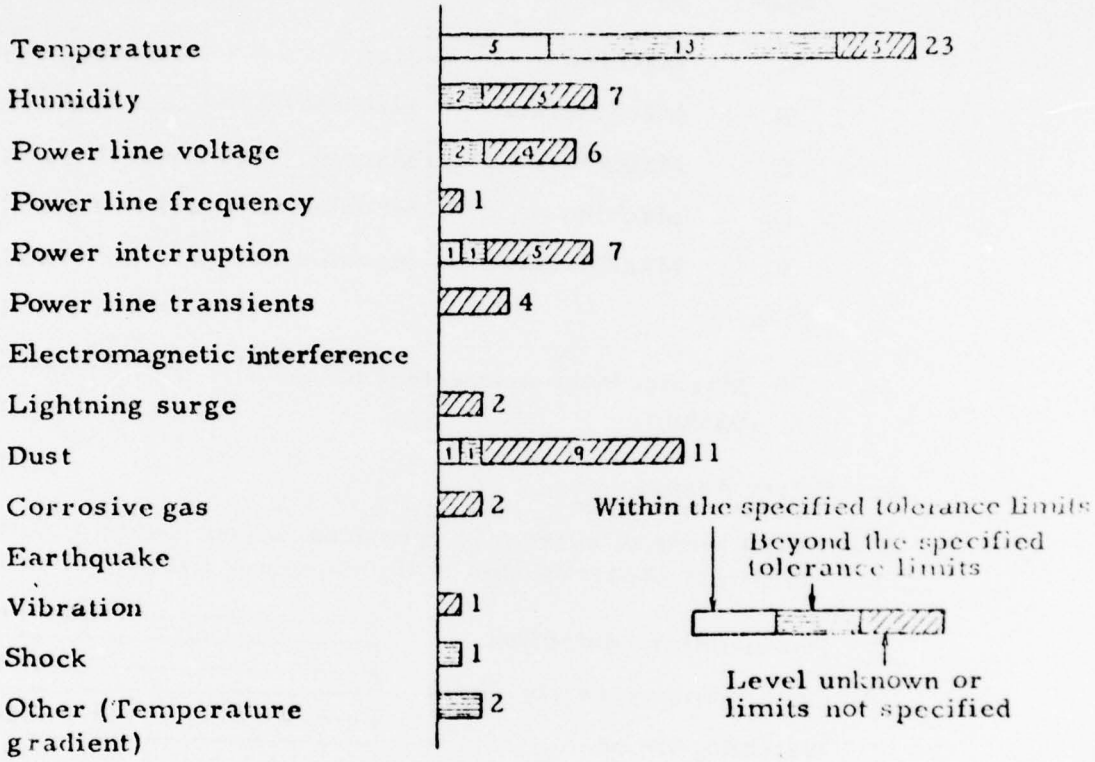
3.5 How long, on an average, was the system down or functioning at a reduced level before one program bug could be corrected?



BEST AVAILABLE COPY

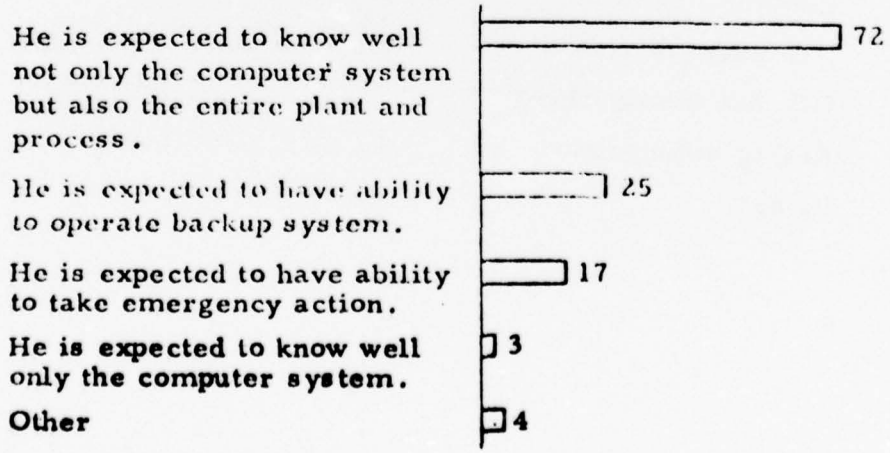
4. Installation Environmental Conditions

4.1 Have environmental conditions caused any trouble in your system? If they have, what was the factor that caused the trouble, and was the condition within the specified tolerance limits?



5. Operator Responsibilities

5.1 What level of ability is the operator in your system expected to have, over and above the ability to perform the usual operations?



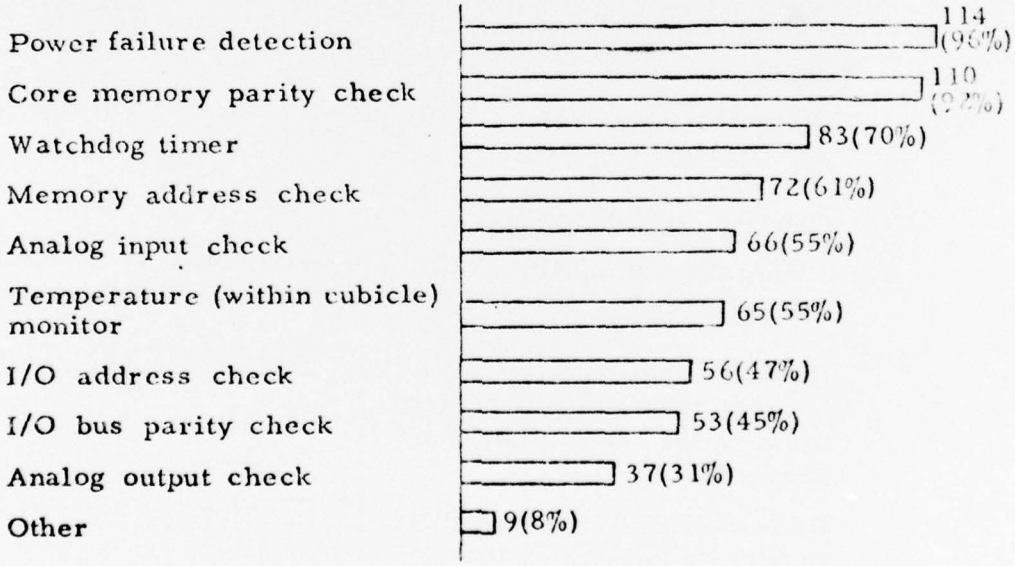
5.2 Which type of sharing of responsibility between man (operator) and machine (computer) do you think most desirable with respect to safety during normal operation? During abnormal operation?

Type	Normal operation	Abnormal operation	
A	Machine	Man	39
B	Machine/Man*	Machine/Man	33
C	Machine	Machine/Man	24
D	Machine	Machine	5
E	Man/Machine	Machine	2
Other			7

* Machine/Man means that machine is master and man is backup.

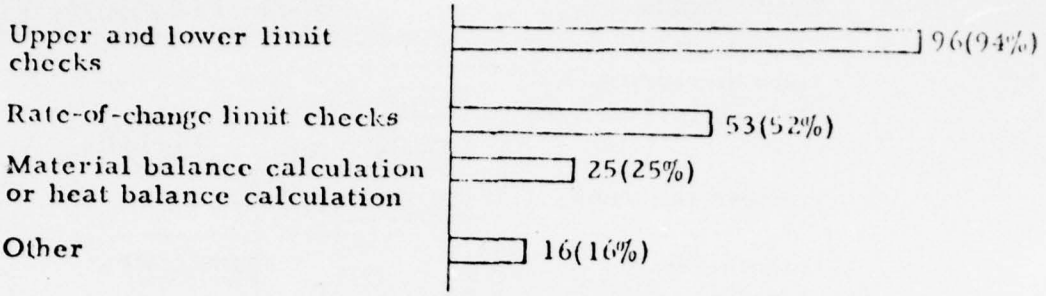
6. Safety Assurance

6.1 What kinds of automatic functions is your system provided with, to detect abnormalities in the computer system?

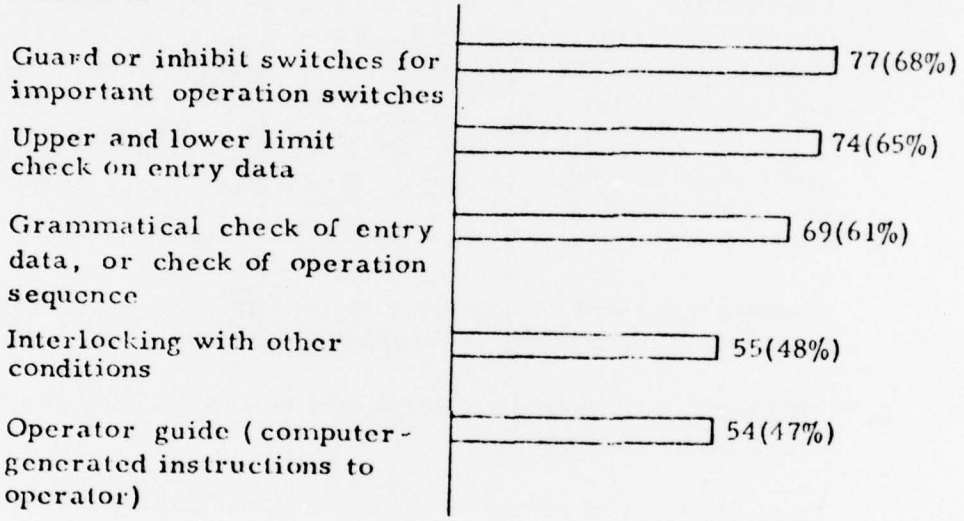


BEST AVAILABLE COPY

6.2 What methods are used in your system to detect process abnormalities?



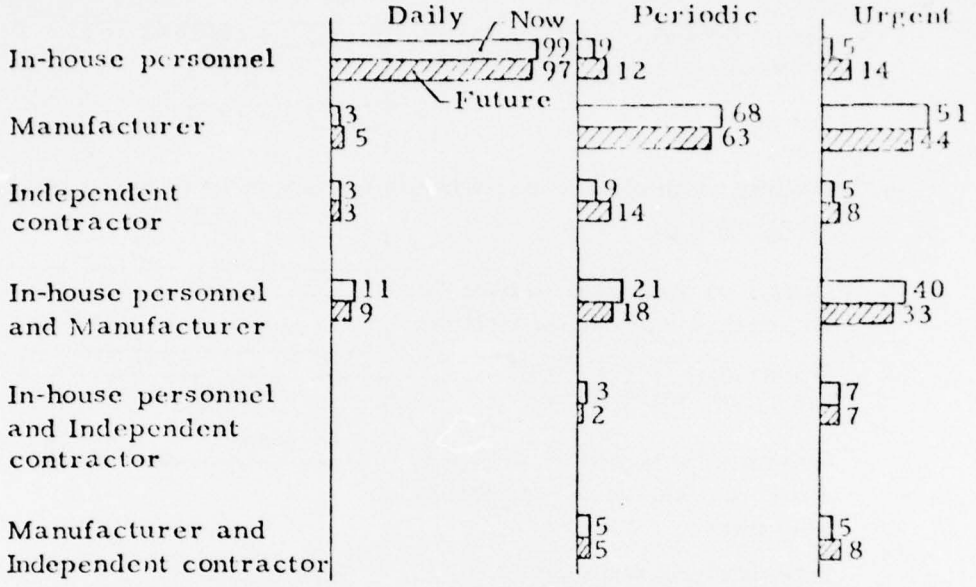
6.3 What methods are used in your system to prevent operator errors?



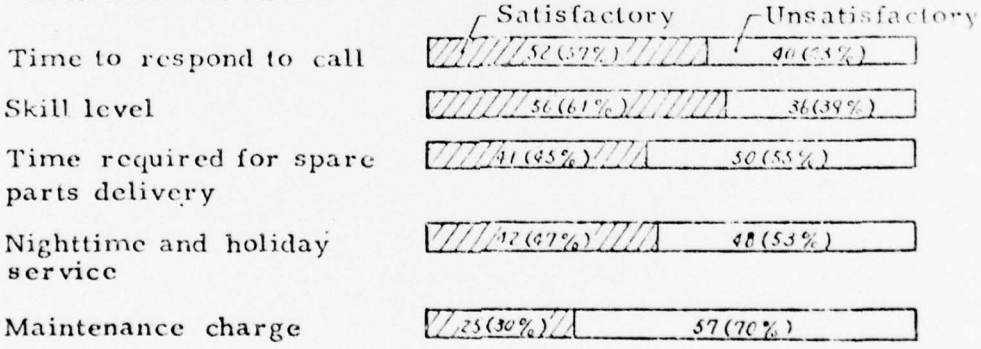
BEST AVAILABLE COPY

7. Maintenance

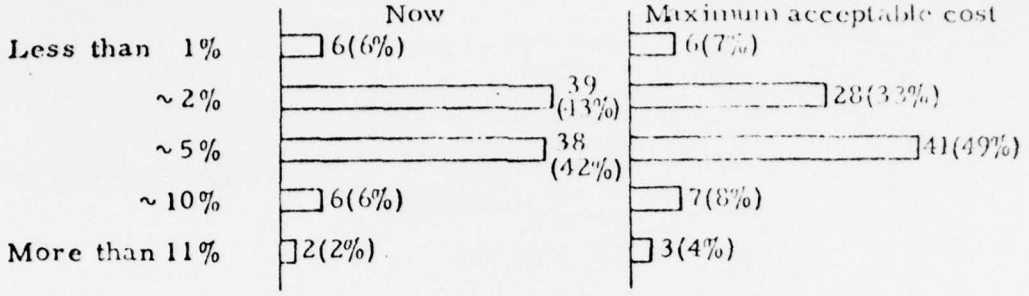
7.1 By whom is computer maintenance (daily, periodic, and urgent) now performed?
How will it be handled in the future?



7.2 If contract maintenance personnel are used, how would you rate their services?

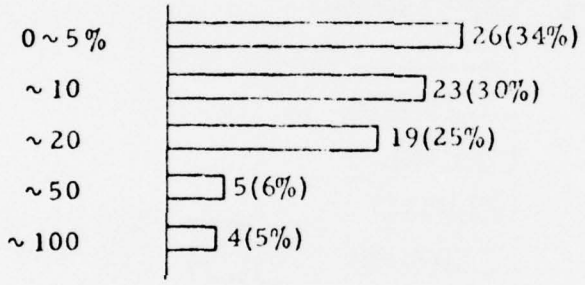


7.3 How much is your annual system maintenance cost (in percent of initial system cost)?
What is the highest maintenance cost that you would tolerate?

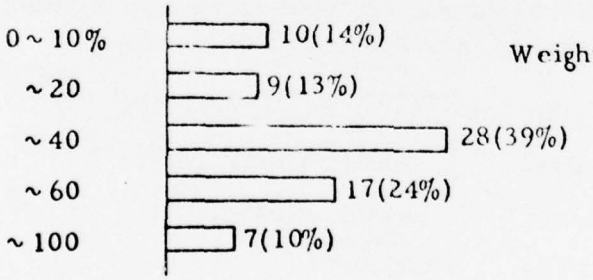


8. Cost/Safety Trade-Offs

8.1 About what was the initial control system (instrumentation including computer systems) cost as a percentage of total investment in plant equipment?

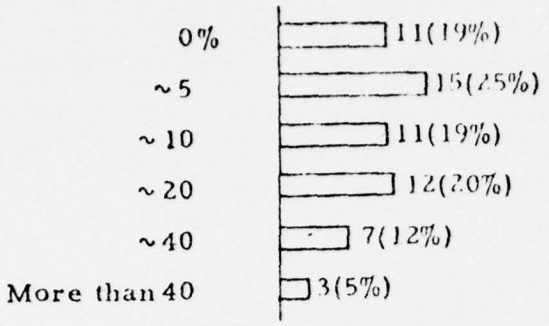


8.2 About what was the computer system cost as a percentage of control system cost?



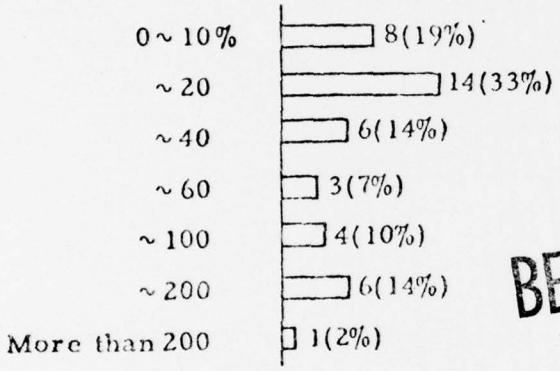
Weighted average: 34.3%

8.3 About what was the backup system cost as a percentage of control system cost?



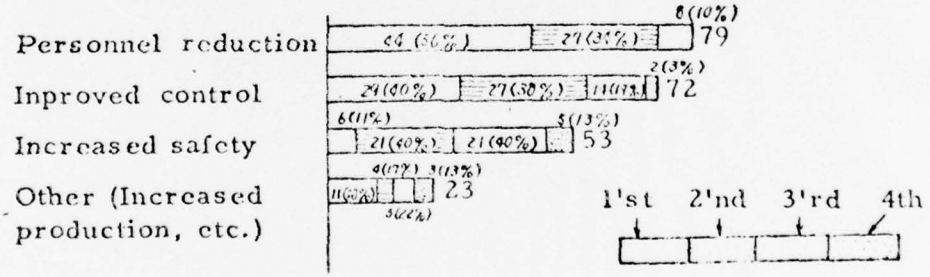
Weighted average: 8.6%

8.4 About what is the annual return obtained by utilizing your system (as a percentage of its initial cost)?

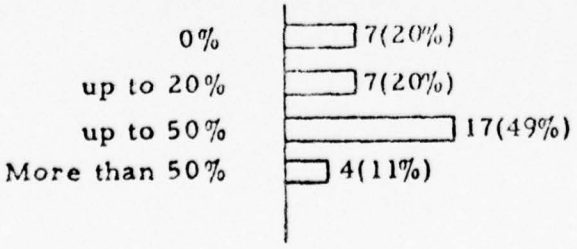


BEST AVAILABLE COPY

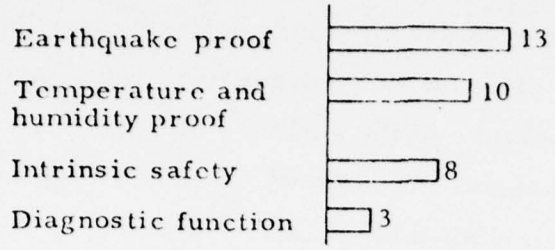
8.5 What benefits have you realized from your system? Rank the benefits in order of importance.



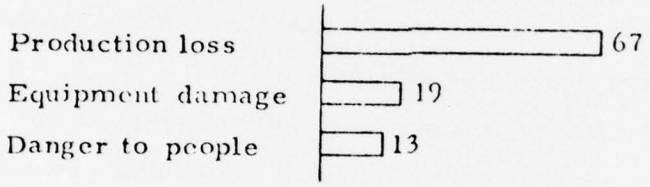
8.6 How much more (than your present system cost) would you be willing to pay for a computer system that would be highly reliable in your application?



8.7 What sorts of functions and capabilities would you require hereafter to enhance safety in your application?



8.8 What effects would there be in case of a complete break down of your system (including backup)?
About how much would the loss and damage be in terms of money?



About 15 users indicated amounts of money for production losses, ranging from 500,000 to 5,000,000 yen (1,700 to 17,000 dollars).

COMMENTS

(1) Though reliability and safety requirements should, of course, be related to system application and scale, the results of this survey give little evidence that a significant relationship now exists, perhaps because of the shortage of samples.

(2) Although techniques for evaluating reliability are now comparatively well established and widely employed by control system users, evaluating safety still presents many difficulties due to the lack of fundamental criteria for evaluation. Check lists for evaluating system safety are therefore urgently needed, and should be developed as soon as possible, in advance of establishing a quantitative evaluation standard.

(3) The questionnaire responses indicate that many malfunctions have resulted from neglecting consideration of environmental conditions, and from incomplete specifications of operating conditions by both users and manufacturers. The effect of some environmental conditions (in particular, power interruptions, power line transients, dust, and corrosive gasses) are inadequately specified (or not specified at all) by manufacturers, due to the difficulties of measuring the effects, and of quantitatively defining acceptable limits for these conditions.

We hope that standards making organizations such as the IEC will actively pursue standardization of these conditions for industrial control systems.

Our regional committee on safety and security (WG-1), took up the problem of installation environmental conditions as the theme of this year's activity, and is now preparing a guideline concerned with operating conditions.

(4) Questions about maintenance were included because of its importance as a factor influencing system reliability and safety, and users responded to them by providing not only the basic data requested, but also extensive comments concerning maintenance documents, maintenance charges, training, spare parts availability, etc. We may gather, from the large number of users responding with detailed comments; and from the nature of those comments, that the majority of users have a lot of problems in maintaining their computer systems.

Thus, our regional committee on safety and security (WG-2), is now investigating maintenance problems and preparing a guideline concerned with computer system maintenance.

PROCESS COMPUTER APPLICATIONS IN THE FEDERAL REPUBLIC OF GERMANY

PAST, PRESENT AND FUTURE OF FUNDED ACTIVITIES

Project Staff-PDV
(Prepared by H. Walze)

Introduction

In July 1975 about 7ooo process computers have been installed in the FRG. The corresponding amount of investment ran up to 1.8 billion German Marks at this time. The recent increasing of installations exceeds 25 per cent per year.

Since application efficiency of this on line computer power was - and still is - unsufficient the FRG Government decided to support appropriate works by a certain part of its funding program.

In 1971 the Federal Ministry of Research and Technology charged the Nuclear Research Center Karlsruhe to coordinate all funded process computer applications in the FRG. For this purpose the Project PDV (Prozeßlenkung mit Datenverarbeitungsanlagen) was founded in Karlsruhe. It is managed by the project leader Dr. Stams and a staff of 5 scientists, 4 assisting engineers, 2 financial experts and some secretaries.

The real work of PDV startet in spring 1972 with the definition of a well established set of guidelines describing short and long term objectives for the project.

In summer 1975 funding contracts for about 15o works in industrial companies, in university- and other institutes were closed.

PRECEDING PAGE BLANK-NOT FILMED

The yearly PDV budget amounts between 21 and 27 million German Marks but the total amount is higher because all commercial companies pay 5o % of their real costs themselves.

In the following an outline of main PDV activities is given.

Man-Machine-Communication

With the increasing use of computer for process control and monitoring, human operator tasks become more and more complex especially when quick reactions are requested. Therefore the adaption of man-machine interfaces to human capabilities is of prime importance.

The following titles of funded works may give an impression of activities in this area:

- Guidelines and specifications for computer aided process control rooms, development of certain devices
- Data presentation schemes
- System development for process control rooms
- Design criterias for command stations in computer based plants
- Process terminal with an integrated minicomputer
- Monitoring equipment for computer aided line switching

Software

Most of the past and recent efforts are dedicated to specification and implementation of the general purpose real time programming language PEARL. PEARL stands for Process and Experiment Automation Realtime Language and was jointly developed by users in industry and research institutes as well as by process computer manufacturers.

Some significant accounts of this high level language are:

- The subdividence in a system-and a problem part makes the usage of programs for different hardware configurations easier. Most hardware dependencies can be put in the system part. In the case of using another type of hardware, only the system part of the program has to be changed while the problem part remains unchanged.

- In order to achieve portability, a standardized interface to the hardware dependent computer operating systems is being defined. This refers mainly to input/output, interrupt handling and tasking.

Besides individual implementations two portable PEARL programming systems are being developed.

One of the portable compilers is supposed to run on small target mini-computers of not more than 48 K core memory.

The following table gives a survey of present implementations.

Computertype	Implementing Institution
AEG 60-10	University of Stuttgart
AEG 60-50	University of Stuttgart
SIE 4004/3	ESG, München, and DFVLR, Oberpfaffenhofen
SIE 3006	University of Erlangen
AEG 80 family	AEG, Konstanz
SIE 330	Siemens, Karlsruhe
DP 1000	Brown, Boveri & Cie., Mannheim
HP 3000	Institut für Rundfunktechnik, München
EPR 1100	Krupp Atlas, Bremen
MULBY 3	Krantz, Aachen
MINCAL 621	Dietz, Mülheim
T 1600	Software Partner, Darmstadt

In national as well as in international expert groups PEARL is considered as serious candidate for a standardized real time language.

Besides PEARL the problem oriented language PSF is under development in PDV-works since 1973. PSF is the short term of Problem-spezifische Sprache für Förderprozesse. It is tailored for discrete particle automation like warehousing or material distribution in industrial production processes.

PSF as a problem oriented language is of higher level than PEARL and is easy to learn for all application engineers in this field. Furthermore PSF is an appropriate tool for planning purposes. It is easy to implement on computers which already have PEARL compilers.

In the present PDV project plan the submission of PSF is envisaged for 1977.

The long term goal for all works in that field provides a common software production system. With such an integrated system the overall cost for software generation can be reduced rapidly.

It will consist of:

- Precompiler for supporting structured programming and decision table applications
- Means for source program handling like editing and macrogeneration
- Means for compilation and interpretation of source programs
- Mounting, linking and loading programs for object codes
- Machinecode generators
- Test means

  etc.

Hardware

The fast progress in the field of mini CPU's and microprocessors does not solve interface- and other problems occuring when computer power has to be integrated in a process control system.

Here the following goals can be derived from the users requirements:

- Special application dependent design cost must be reduced
- Commissioning and installation cost must be reduced
- The system must be more available and more reliable
- Operation and maintenance features must be improved.

The global approach for a solution is: Intelligence distribution. Therefore many PDV-works contribute to design, development and implementation of modular decentralized systems with standard interfaces.

The common base of these developments is a bit serial line sharing system - the PDV-Bus - with uniform interfaces to all attached stations. It makes start up and maintenance much easier and enables compatibility between system components from different sources.

The PDV-Bus concept is being developed since spring 1974. It is now in the phase of implementation. The mandatory part of the bus specification comprises communication protocols and hardware interface functions.

Characterizing features are:

- Hierarchic structure based on the master-slave scheme
- Slavestations with and without demand capabilities

- Communication path without active components enables passive bus couplers (T-Junction).
- Procedures with command - reply sequence (commands from master to slave(s) and replies from slave to master)
- Transmission of single 16 bit data or data fields with variable length (n x 16 bit)
- 252 slavestations in maximum
- 8 bit CRC field always checks 16 preceding information bits
- Asynchronous demand requests
- High efficient status polling procedure
- Direct communication between 2 slavestations as option
- Between busline and slavestation all informations are transferred as NRZ signals accompanied by separate clock pulses

Eight basic types of transmission procedures are defined:

- Global commands
- Individual commands
- Writing of single data (master ⟶ slave)
- Writing of data fields
- Reading of single data (slave ⟶ master)
- Reading of data fields
- Reading and writing of single data
- Reading and writing of data fields

First experiences with the PDV-Bus are expected for 1977.

The system proposal was also submitted as working paper to the Technical Committee on Interfaces and Data Transmission (TC 5) which is part of the European Purdue Workshop on Industrial Computer Systems.

For the PDV-Bus appropriate control- and monitoring equipments as well as DDC- and CNC slavestations are being developed. Many of them are based on microprocessores.

As an alternative for the present analogue and the future digital field instrumentation components using frequency-analogue signal notation were developed.

Their advantages are

- Signal structure permits simple and inexpensive conversion in digital values (using simple counters!)
- High noise immunity (comparable with digital notation) can be achieved
- High accuracy allows applications in dosing processes like weighing etc.

Certain prototypes are partly under test in industrial plants and the wide commercial introduction is intended. Nevertheless the step to common industrial use is not yet done.

For ACO (Adaptive Control Optimization) machine tool control systems special sensors with high accuracy and sensitivity were developed. In detail sensors for tool wear, for the appreach of tool to the material, for cutting force and for material surface roughness are available. Application areas are cutting processes like turning, drilling, milling and grinding. These works were performed in several university institutes being coordinated in the so-called HGF (Hochschulgruppe Fertigungstechnik).

Another important PDV subject are two methods for automatic diagnosting of hardware faults in computers. One method makes use of the "none exact dictionary matching", the other evaluates logical bit sequences appearing at a certain time at different test points.

Compared with conventional diagnosting techniques these advanced statistical methods operate with a smaller data base and are widely computertype independent. They are implemented for an AEG 60-10 computer as test object. The diagnosis is controlled and evaluated by an external test processor realized with an AEG 60-07 (Interdata-CPU).

Pilotsystems

The third class of PDV-works are exemplary control computer applications in certain industrial branches.

The most important representative is the FFS (Flexibles Fertigungssystem), an integrated system for computer managed parts manufacturing. The FFS will be realized in a cooperative manner between a university institute and a big manufacturer for Diesel engines and parts of them (MTU).

The system is expected to automate metal cutting- and transport processes in a MTU factory.

Nearly ten different types of engine parts like cylinder blocks will be produced on six numerical controlled machine tool centers with automatic tool changing and load/unload devices. All machine tool centers are linked together and with a store system so that all parts and tools can be transported by computer control.

It is intended to integrate the FFS into the current MTU production in 1979.

A second pilotsystem which should be mentioned was designed by BFI (Betriebs-Forschungsinstitut) for computer-controlled steelmelting in an electric arc furnace.

Here the energy distribution control system monitors the electrical power consumption and allots the disponable melting power to the different furnaces. This happens in accordance with their priorities at a certain time.

The comupter based furnace control additionally records the charging run off and optimizes the electrical arc operating parameters.

The attached PDV publication list reflects past and present topics of funded activities (it is only a selction).

All KFK-PDV reports are available from

Projekt Prozeßlenkung mit DV-Anlagen (PDV)
Gesellschaft für Kernforschung mbH
Postfach 364o

75oo Karlsruhe
Germany

KFK-PDV 1

K.H. Timmesfeld, B. Schürlein, P. Rieder, K. Pfeiffer, G. Müller, K. Kreuter, P. Holleczek, V. Haase, L. Frevert, P. Elzer, S. Eichentopf, B. Eichenauer, J. Brandes:

PEARL - A proposal for a process- and experiment automation realtime language

KFK-PDV 3

R. Werthmann:

Decision table preprocessor for control systems (system design)

KFK-PDV 5

H. Herbstreith, P. Kossmann:

Performance characteristics of today's process control computers

KFK-PDV 6

R. Werthmann, E. Kazis:

Interactive dialog system

KFK-PDV 1o

P. Namneck, I. Schnarre, K. Wagner:

MULI - Multi Level Dialog System,
Multi Level Dialog Language - description of language and system

KFK-PDV 11

P. Namneck, I. Schnarre, K. Wagner:
MULI - Multi Level Dialog System,
Multi Level Dialog Language - case studies for application

KFK-PDV 12

E. Kazis:
Interactive Dialog System Part II

KFK-PDV 13

W. Kutzsche, U. Voges:
GAUSS - a System for Structured Programming

KFK-PDV 14

H. Unbehauen, B. Bauer, B. Göhring, Chr. Schmid:
"On line"-Identification Methods
Evaluation of Literature and Review of the Methods

KFK-PDV 17

Rudi F.W. Grimm:
Comparison of Redundancy Reduction Algorithms with respect to System Protection Using Process Controllers

KFK-PDV 21

Hans D. Hoelzer, H. Röck, J. Waidelich:
Methods of detecting and locating faults of combinatorial and sequential circuits

KFK-PDV 26

H. Unbehauen, F. Böttiger:
Control algorithms for implementation on process control computers

KFK-PDV 28

E. Verhaag:
Machine tool control, computer linked numerical control

KFK-PDV 3o

F. Freyberger, G. Landvogt, G. Schröder, H.R. Tränkler:
Use of Frequency-Analogue Signalnotation for Computer Aided Process Control

KFK-PDV 33

G. Färber:

Redundant Serial Loop-System with standardized Interfaces

KFK-PDV 37

H. Unbehauen, Chr. Schmid, F. Böttiger, B. Bauer, B. Göhring:

KEDDC: A combined process computer program system for the design and application of DDC algorithms

KFK-PDV 38

R. Zimmermann:

Design of man-machine-communication-systems

Basic human factors, demands and recommendations

KFK-PDV 41

K. Essel, W. Hänsel:

Development of sensors for process control systems in the field of production engineering

KFK-PDV 42

R. Hoefert, J. Lemmrich:

Compilation, classification and evaluation of components for frequency analogue process instrumentation systems

KFK-PDV 44

H.J. Günther, W. Werum, H. Windauer:

Problem oriented language for transport systems, language report

KFK-PDV 47

W. Patzelt, M. Salaba:

A method for performance comparison of DDC algorithms and application of this method to selected cases by means of the computer program OPTAL

KFK-PDV 5o

H. Birk, G. Färber, H. Halling, D. Heger, M. Patz, E. Holler, H. Walze:

"Mikroprozessoren und Mehrprozessorsysteme für die Prozeßlenkung"

KFK-PDV 53

F. Wagner, H. Woda:

"Process Basic"

KFK-PDV 54

R. Isermann, D. Bux, P. Blessing, P. Kneppo:

Control Algorithms for direct digital control with process-computers

KFK-PDV 56

SCS, Scientific Control Systems GmbH:

MULI - Mulit level dialog language - language and system description

KFK-PDV 59

H. Siefkes:

A microprogrammed process unit

KFK-PDV 6o

R. Güth:

Microprogrammable Interface

ALUMINUM COMPANY OF AMERICA
ALCOA BUILDING · PITTSBURGH, PENNSYLVANIA 15219

ALCOA

January 23, 1973

Dr. T. J. Williams
Purdue Laboratory for Applied
Industrial Control
Purdue University
Lafayette, Indiana 47907

Dear Ted,

Enclosed are the diagrams I promised you, in connection with the ISA workshops.

Figure 1 shows some representative process time constants. These figures, even for a given type of control loop, such as a pressure control, vary over quite a wide range, due to system capacities, flow rates, and available energy. Thus, the pressure in a large reactor may exhibit a response time of the order of several seconds, while a hydraulic servo positioner such as an antenna control system may operate in the millisecond range. Because of considerations like this, I shudder whenever someone generalizes to the extent of saying that, in a ddc system, "pressures should be sampled about once per second". Indeed, in the antenna positioning example, the pressure control loop may be entirely inside a speed control loop, which is in turn inside the position loop of interest. In this case, maybe the position can be sampled about once per second for adequate control, but the speed and pressure loops would operate at increasingly faster sample rates.

Cascaded controllers with adaptive features further complicate things, because inner loops may be directed by the dynamic behavior of outer loops, and sample times can be adjusted as part of the control strategy, on top of more conventional practices of adjusting set points or providing compensation for measurements.

If workshops are to produce guidelines for system design activities, some of these guidelines may well have to be dimensionless in order to be useful. An example is the "4 samples per time-constant" rule, which can be restated as the ratio of sample time to response time (1/16 in this case).

Some relative ranking of the dynamics of nested control loops is also useful, such as, in the case of strictly linear loops, that each succeeding inner loop (of a stable system) will generally respond about twice as fast as the loop around it. This can be carried further, for non-linear systems, by saying that the extent to which this "rule of 2" is violated is a measure of either non-linearity, instability, or sluggishness.

PRECEDING PAGE BLANK-NOT FILMED

Figure 2 continues the idea of structuring generalizations into a useful form, on a dynamic scale. The lower four (4) curves represent typical ranges of regulators and sequencing control systems as used in industrial manufacturing operations. There are exceptions to these ranges, of course, but it is my experience that the majority of control applications fall into these ranges. Further, the relative differences between the control technologies shown appear to be real.

The big jump shown for automatic multivariable control is due partly to the fact that such systems rapidly become quite complex and tend to be digital, with memory that is adequate for the handling and correlation of extremely slow variables. (A human operator, attempting to operate over this wide a dynamic range, has difficulty remembering enough information to form the necessary correlations, and he also has trouble observing rapidly changing variables and responding to them.)

The solid bars from the fifth level upward represent the typical dynamic ranges of production monitoring, production control, and general business information systems, and are designed to aid people at various levels in a business management hierarchy. The dynamic ranges of the people themselves, at various levels of the hierarchy, are represented by the dashed horizontal bars near their corresponding information system graphs. Since the information system used by a person at a given level is part of his "process sensor" system, it must be capable of faster dynamic performance than the person who is acting as the controller.

In a structure such as this, it must be remembered that some activities are not represented by the structure, and that the information and control systems and the people sometimes operate outside the dynamic ranges shown. Thus, if the plant manager wants a pencil sharpened, he can certainly sharpen it himself in less than half a day, but if he delegates the job down to a production or maintenance worker, it may very well take that long. The chart of Figure 2 is intended to show the relative dynamics of tasks that operate by delegation down thru the organization, related to the hierarchy of people and information systems that make up this organization.

It has been known for some time by control system engineers, that they must do a thorough job of human engineering in order to achieve a satisfactory level of a quality known as "operator acceptance," to make a system perform its functions as intended, and to achieve the greatest benefit for the engineering work that a system represents. I have observed that people of all levels in a hierarchy tend to tailor their jobs until they achieve some level of personal satisfaction, or "comfort." The diagonal shaded stripe of figure 2 represents the approximate locus of such "Comfortable Operating Points" (COP) for people at various levels.

If a man is too busy (or uncomfortable) at a task that is quite dynamic, he will tend to ignore some requirements until he achieves a comfortable operation, that he can handle to his own satisfaction. This sort of operation may not result in ranking of priorities in the same way these priorities might be ranked by the man's supervisor, though, so the result may be organizational conflict.

On the other hand, if a given job is too routine, with "nothing happening," a man tends to make work, of his own choosing. Some of this made-work can be of benefit, but the yield in this respect, as far as the overall organization is concerned, is apt to be low if such activities are without purposeful direction.

It appears to be a valid system engineering strategy to recognize the significance of the COP's of the people who are to build, operate and maintain systems, regardless of their relative hierarchical level, and to design these systems in such a way that the human organizations mesh constructively with the resulting organization of mechanisms and information systems. Industrial psychology, with its emphasis on job enrichment, is part of this. So is a thorough knowledge, on the part of the systems engineer, of the various organizational objectives and the personal objectives of the people involved, plus their relationship to his own objectives.

The integration of man/machine organizations also involves the observation of biological constraints, as depicted by the vertical line marked "$\alpha = 0.007$ sec," which marks the area of human mental operations. Sequential mental processes seem to be limited here, so a person's ability to observe a process, correlate the observation with a pattern of desired process behavior, and take action has a definite break frequency. The system engineer, through knowledge of these kinds of limitations, may be able to design operator interfaces in such a way as to take advantage of the human capacity for parallel processing (pattern recognition), to perform more elaborate correlations, and to eliminate the need for a large number of sequential thought processes.

At the other end of the spectrum, a person's memory is finite, and organizational jurisdictional boundaries form constraints to human band width at various levels of the hierarchy.

Note that this representation does not account for such phenomena as local motor control (in the human body), where the thought processes operate in a supervisory mode, and the limbs perform high-speed, usually repetitive and complex functions by "reflex action". These functions involve training in manual dexterity and coordination, and are perhaps physically demanding but do not involve mental fatigue. Such functions may legitimately be designed into a job, as in the case of bilateral servos employing force-feedback, to take advantage of the human ability to sense and control tactile phenomena by motor control without conscious thought. An experienced motor vehicle operator, driving along a freeway, operates mostly in this mode.

The vertical scale of figure 2 is not labeled, but I suspect it has something to do with the impact of decisions made at various levels of the hierarchy, with the impact increasing as the top of the hierarchy is approached. Impact, in this sense, is a measure of the futurity of decisions, and a high-impact decision cannot easily be reversed or corrected. Some of the differences in impact are thus due to the inertia of the hierarchy itself, but ideally most differences would be due to deliberate design of a management structure: the division of management responsibilities to correspond with experience available.

Besides the dynamics and impact attributes of jobs, there are many other factors that go into the design of hierarchies, such as freedom of action, know-how, the nature of interpersonal relationship, the geographical distribution of functions, materials and information, training costs, etc. A systems engineer certainly has a lot to consider, if he is to successfully design elaborate hierarchies of man/machine systems. I suggest that the simplest strategy for making the dynamic/impact tradeoffs would be to attempt to structure the jobs of a hierarchy so that the locus of all jobs lies roughly along the COP locus (rather than horizontally, as shown), so that each job has a mix of both impact and dynamic content, and so jobs effectively form a continuous spectrum through the chain of command, with each job covering a dynamic range of about two (2) decades.

On the mechanical side, the geographical distribution and task breakdown of an hierarchical system of control equipment does not lend to simplistic solutions either. Much of the literature on hierarchical system philosophies has attempted to arrive at neat divisions of tasks that lend, in turn, to neat block diagrams. Much consideration must be given to other, more important factors, such as: 1) matching built-in isolating effects, such as queueing areas or storage capacities of a process; 2) providing for fault isolation; 3) allocation of controller loading; 4) distribution of information sources and sinks; 5) availability of communications resources; 6) economics of scale; 7) economics of specialization; and of course, 8) the nature of the human organization.

Arbitrarily limiting a local ddc controller to only the last loop, or last few loops of a cascaded system is apt to lend to poor economics in the use of digital control. Manufacturing control extends from the marketplace to the Boardroom, and on down to the process, and can contain dozens of levels. Division of these levels into suitable hardware hierarchies is a first-class systems design problem, and should not be glossed-over with simple philosophies that make neat block diagrams.

R. L. Curtis

FIGURE 1 --

Process Time Constants, Seconds

R. L. CURTIS
-- ALCOA

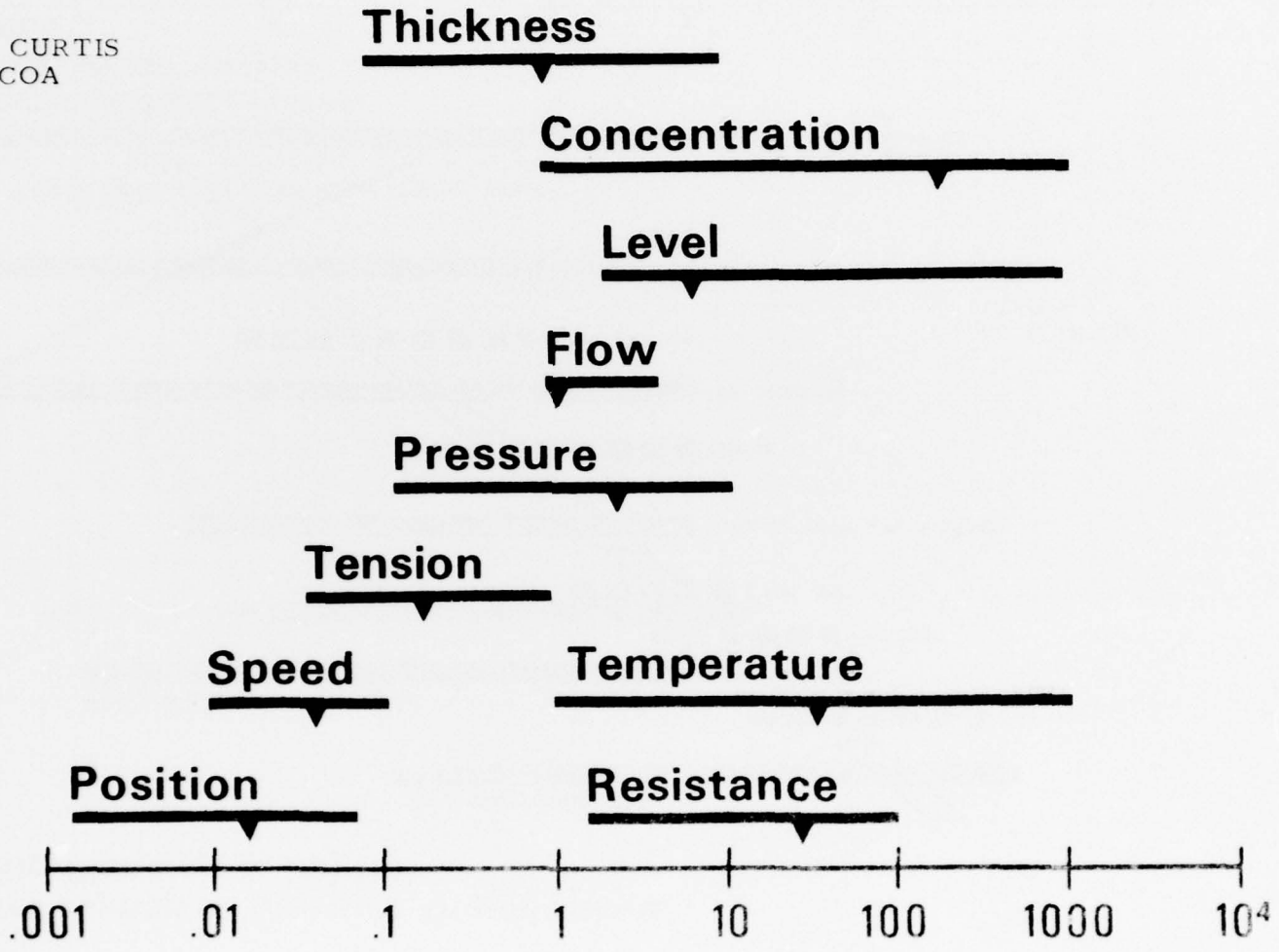
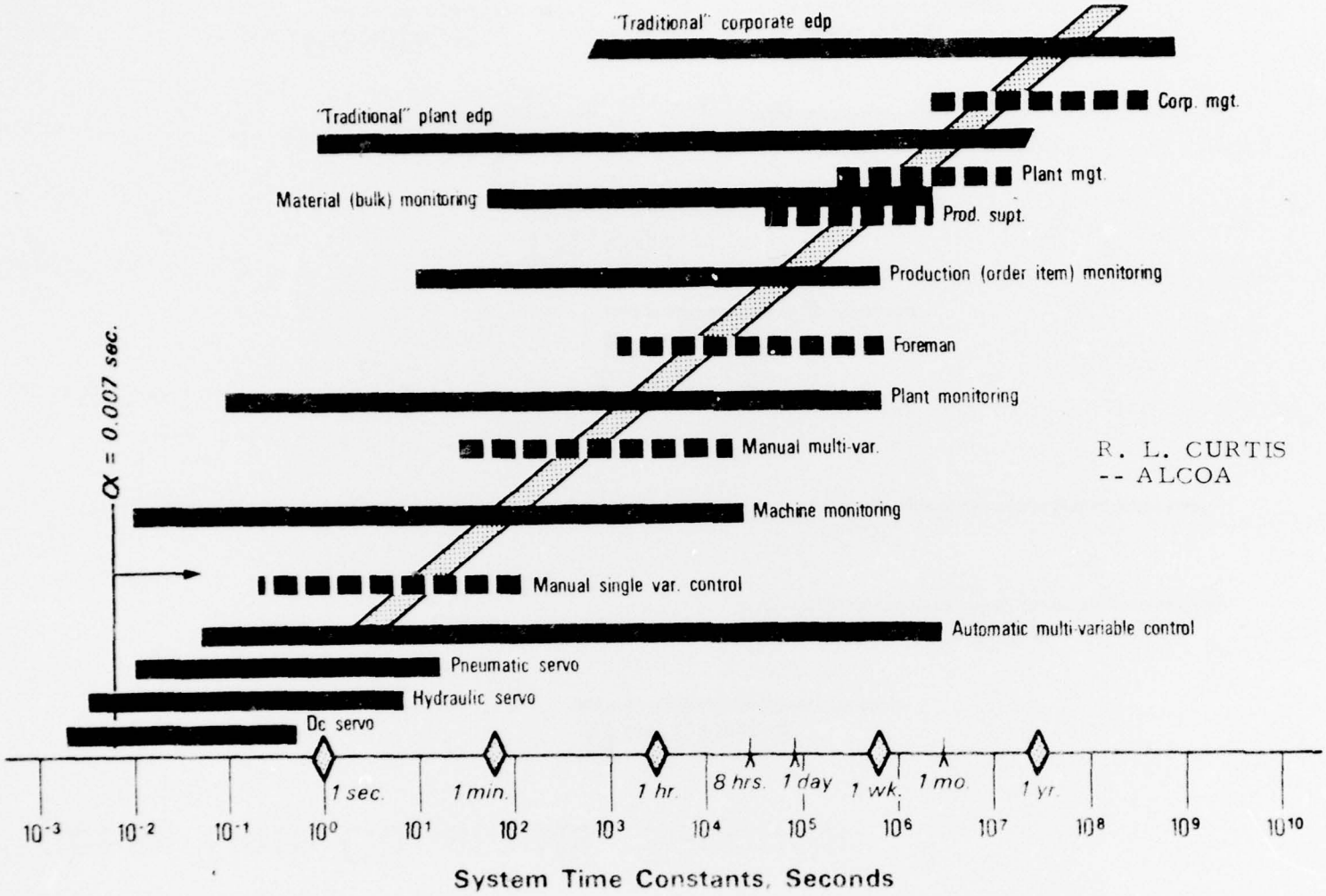


FIGURE 2 --

The Dynamics of Human & Machine Organizations In Manufacturing Planning & Control



R. L. CURTIS
-- ALCOA