MASTER

IS-2228

Physics (UC-34)
TID-4500

UNITED STATES ATOMIC ENERGY COMMISSION

Research and Development Report

AMES LABORATORY ON-LINE CONTROL
SYSTEM FOR BUBBLE CHAMBER
FILM MEASURING

by

W. J. Higby and A. H. Klein

December 1969

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Ames Laboratory
at

Iowa State University of Science and Technology
Robert S. Hansen, Director
Contract W-7405 eng-82

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

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

DISCLAIMER

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

IS-2228

This report is distributed according to the category Physics (UC-34) as listed in TID-4500.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.65

TABLE OF CONTENTS

	Page
ABSTRACT	v
INTRODUCTION	1
SOFTWARE FEATURES	3
HARDWARE FEATURES	4
JOB SCHEDULING AND TIME SHARING	5
ADAPTABILITY TO OTHER INSTALLATIONS	8
KUBER ON-LINE CONTROLLER	8
BASIC DESIGN PHILOSOPHY	8
INTERRUPTS AND JOB ACTIVITY STATUS	9
REFERENCE TO COMMON VARIABLES	11
THE BASIC KERTRAN STATEMENTS	14
MEASURING MACHINE COMMANDS	23
CONCLUSION	31
ACKNOWLEDGMENTS	32
REFERENCES	32
INDEX	33

IS-2228

IS-2228

AMES LABORATORY ON-LINE CONTROL
SYSTEM FOR BUBBLE CHAMBER
FILM MEASURING

W. J. Higby and A. H. Klein

ABSTRACT

This report discusses the bubble chamber film measuring control system used at Iowa State University. The main features of the system provide for complete flexibility of control criteria used for a given experiment. The syntax of the KERTRAN language is given with examples of how it is used for control purposes.

IS-2228

INTRODUCTION

A system for on-line computer control and analysis of bubble chamber film measurements has been developed by the High Energy Physics staff at the Ames Laboratory. The overall criterion placed on the design of this system was the necessity for complete flexibility. The ground rules are:

- 1) The system must be capable of operating with totally different processes on different measuring machines on-line.
- 2) The programming involved to define some measuring process must be of such a nature that it is convenient for any staff member to do.
- 3) Total geometric reconstruction of events measured should be done on-line.
- 4) Some capabilities for kinematic fitting of the track measurements should be possible on-line.

Considering the first two ground rules, the system configuration operating with measuring machines on-line could be as shown in Figure 1.

A physicist or graduate student who is doing some experiment on-line decides what functions his particular process will perform and the quality control criteria to be imposed on the measurements. He need not concern himself with details such as which measuring machines he will be using and what processes other machines will be performing. The description of the measuring process is written in a language quite similar to FORTRAN. We designed this language with the thought in mind that most people involved in High Energy Physics are familiar with FORTRAN and the statement structure is quite adequate to define a measuring process.

FIGURE 1. A possible system environment.

Since the system totally reconstructs each event in space as it is being measured, an extremely high quality control of the measured data is achieved. Not only can one check for track failures per view, but also tests on momentum errors of the completely reconstructed track can be made.

An on-line version of the kinematics routine GUTS is available for use as another high quality control function. We have found from past experience that on-line kinematic fitting is extremely useful in experiments that involve the measurement of events with Vees. It is also useful in any experiment where nearly 100% efficiency of data collection in a particular final state is desired.

SOFTWARE FEATURES

The basic language of the system is KERTRAN, Kuber Expression TRANslator. Programs written in this language execute the various measuring machine functions such as lighting a view lamp, displaying a message to the machine operator, reading a data point, etc. In addition, they may call for major routines of the system to be executed. The major system routines include the reconstruction programs, kinematics routines, and various other programs of the system. The KERTRAN compiler¹ is written in FORTRAN so that if other installations use our system they will have the freedom to modify or add to the language as they desire.

The actual instruction code generated by a KERTRAN compilation is not directly executed by the hardware of the E.M.R. 6050 computer. The general controlling program KUBER² is, in a sense, a central processing unit simulator. KUBER treats the compiled KERTRAN code as a sequence of data instructions

which it interprets and executes. KUBER also processes all hardware interrupts from the measuring machines, and controls job scheduling and time sharing for the system.

HARDWARE FEATURES

The basic types of information transferred between the measuring machines and the computer are

1. Encoder and Button Settings -- three words of data that are read from the machine by the computer.
2. Message Display -- Alphabetic messages displayed on the measuring machine message display section.
3. Binaview Display -- A series of 15 numeric displays, each of which may display any digit.
4. Interrupt Buttons -- Two buttons which cause distinct hardware interrupts.

The message display section is a means of causing one of 36 messages to be displayed to the operator. These messages are usually used to inform the operator of some geometric failure detected during the measuring process. The Binaview display communicates only numeric information to the operator such as frame number, event type, grid location, etc. Information from the measuring machine to the computer consists of three words of data that can at any time be read by the computer. Two words contain the present x and y machine encoder values and one word contains the present settings of the operator's button board. Two buttons on the button board panel cause distinct hardware interrupts when depressed. The convention has been to label these 'ACKNOWLEDGE' and 'ENTER.'

JOB SCHEDULING AND TIME SHARING

During the process of measuring an event the possible states of calculation for a given measuring machine are:

1. Execution of KERTRAN code
2. Execution of a LINK
3. Waiting for an Interrupt.

All the major computational programs of the system are LINKS. Each LINK executes a specific function, e.g., process all fiducial measurements in a specific view, process a vertex measurement in all views, reconstruct a track in all views, etc. All LINKS are stored on a high speed random access device. When a KERTRAN program executes a LINK call, the specified LINK is loaded in the LINK Area, as shown in Figure 2, and executed.

KUBER considers the total logic of a process for controlling a measuring machine to be a 'JOB.' In addition to measuring machine JOBS, KUBER will also execute background jobs. Each JOB eventually reaches a state of waiting for an interrupt. For example, JOB Number 1 displays a message to the operator at measuring machine 1 that informs her that the fiducials measured in a specific view have failed the accuracy test. The job must logically wait until the operator looks at the message, then responds by depressing the ACKNOWLEDGE Button. This action informs the system that the operator understands the message and the JOB may continue its operation. During this Wait Period KUBER will execute other jobs that are not also waiting for an interrupt.

FIGURE 2. Memory layout.

Job scheduling is done utilizing a software activity clock. Each time this scheduler is entered, the clock is incremented and the status of the particular job is tested. If the job is not active, the clock is incremented and tested again. If a job is found that is active, KUBER will schedule it for execution. A job

is in the deactive state when it is waiting for an interrupt and is activated when the interrupt occurs. If the Job Scheduler finds that all jobs are not active, it will schedule the system job 'LAZY' to be executed. This Lazy Job is nothing more than an infinite loop which waits until an interrupt occurs for some job. Our past statistics have shown that with three machines on-line³ doing kinematic fitting in addition to the total reconstruction of the events, the Lazy Job is being executed approximately 85% of the time.

ADAPTABILITY TO OTHER INSTALLATIONS

An installation that does not use an EMR 6000 series computer will have to rewrite the assembly language sections of this system. All of the reconstruction and kinematics links have been written in FORTRAN. The only major assembly language routine is of course the KUBER on-line controller. This logic was designed and flow-charted before coding began, so one should be able to use this design and code the controller in any assembly language quite easily.

Our hardware environment consists of a 16K 24-bit word length computer with a cycle time of 1.9 μ sec. The mass storage device is a 5 megabit drum with a transfer rate of 96 K words/sec. Event measurements are recorded on a nine-track IBM compatible tape drive. Other I/O devices are a 200 card/minute reader and a console Teletype. We feel that this is a minimum configuration for optimum execution of the system. The tape drive could be a seven track in place of the nine track.

KUBER ON-LINE CONTROLLER

BASIC DESIGN PHILOSOPHY

The core resident controller, KUBER, is a CPU simulator that executes a sequence of instructions known as a task word table (TAWTA). An instruction may be a simple task such as load a register, store a register, etc., or it may be rather complex such as displaying a message on the operator message display board. The TAWTA consists of instructions that can not be executed by the CPU, but they are interpreted and executed by KUBER.

The principal advantage of such a scheme is that KUBER has complete control over what TAWTA to fetch an instruction from and execute and when

to fetch the instruction. If an installation had four measuring machines, each of which was assigned to a completely different measuring process, four TAWTA's could be controlling the machines. A possible time-sharing algorithm for such an environment could be a "round robin" one in which an instruction is executed from the TAWTA for machine 1, then one for machine 2, etc. Another advantage of this approach is that one can design what the format and type of instructions are to be.

INTERRUPTS AND JOB ACTIVITY STATUS

The process of executing a given TAWTA is termed a Job. KUBER maintains a Job Activity status switch for each job being serviced. A Job is either active or not active. The scheduling logic that selects a Job for execution of an instruction will not select a Job that is not active.

This technique is used to hold the execution of a TAWTA in a wait state until the operator causes an ACKNOWLEDGE or ENTER INTERRUPT. This action may then activate the job for further execution. An example of the use of this structure is the following process for a measuring machine:

1. Display a message to the operator for a vertex measurement
2. Begin a loop to 7 for all views
3. Turn the view lamp on
4. Wait until operator depresses ENTER and then continue with next step
5. Read the x,y encoder setting of the measuring machine
6. Store the vertex measurement in the appropriate common array
7. Do loop continue statement
8. Call a reconstruction link to process this vertex measurement.

During the wait period at step 4 the operator positions her reticle to the desired vertex point. The action of depressing ENTER then informs the process that the encoder settings can now be read. KUBER is not actually waiting during this period, but is executing other jobs in the system. The KERTRAN source code for the above process would look like:

```

C      MESSAGE 18 IS 'VERT'
      NVERT=1
      DISPLAY MSSG 18
      DØ ST70 NVIEW=1,3
      LIGHT VIEW NVIEW
      PRØCEED TASK,IGNØRE
      IN DAT
      IVERTX(NVIEW,NVERT)=MMX
      IVERTY(NVIEW,NVERT)=MMY
ST70  CØNTINUE

C      CALL RECONSTRUCTION RØUTINE

C      CALL LINK 4
      .
      .
      .
      .
      .

```

The proceed statement has two parameters which specify what action KUBER is to take when an ACKNOWLEDGE or ENTER interrupt occurs for the measuring machine assigned to this TAWTA. The action taken is either to ignore the interrupt (operator has pushed the wrong button) or to activate this job. The following logic flow basically describes the KUBER operation in this circumstance.

The logic block that selects a job that is active was briefly described on page 7 of this report.

REFERENCE TO COMMON VARIABLES

The bulk of the computing load done by the system is not in the KERTRAN control TAWTA's, but is in the reconstruction and kinematic links. Information must be passed from the KERTRAN routines to the Links and vice versa.

All transfer of information is done via common storage. There are four classes of storage available to a KERTRAN program.

Class 0:

This class consists of all the variables that appear in the FORTRAN common statements. Some arrays are indexed by the machine number. IVERTX(I,J,K) is an array that contains the encoder x value for view I, vertex number J, for machine number K. When using this array in KERTRAN, the reference to the machine subscript may be dropped, e.g., IVERTX(I,J). All arrays that are machine indexable have the machine subscript last, so that this subscript may be dropped in the KERTRAN coding.

When KUBER is compiled, a table is included that contains the necessary information describing all the Class 0 variables that the KERTRAN programs may reference. An entry in this table will contain

- a) number of subscripts in the array
- b) dimension limits of each subscript
- c) flag to indicate that last subscript is a machine subscript
- d) core address of first element of array.

With this information available to KUBER, no dimension information is necessary in a KERTRAN program. The only information needed in the KERTRAN code is the table entry number of the array referred to.

Class 1:

This class is used for the majority of non-subscripted variables used for communication purposes in the measuring process. It generally consists of variables such as current view being measured, current vertex number, total tracks to measure as specified by the scan card, etc. It can also be used to transfer information from one KERTRAN

routine to another, and for working storage of variables used in a KERTRAN routine. Variables in this class can be either non-subscripted or singly subscripted.

Class 2:

This class has the same characteristics as Class 1. Most of the information in this class is information read from a scan card for the event to be measured.

Class 3:

All non-subscripted and singly subscripted variables that are local to a specific KERTRAN routine are of this class. Local variables are those that have not been declared as being Class 0, 1, or 2.

These variable classes will be discussed in more detail in the section that covers the KERTRAN language.

The variable convention described here is not an unchangeable part of the system. It works well for our usage, but another installation may desire to modify the convention we use. A section of KUBER that processes variable addressing could be changed with very little difficulty.

THE BASIC KERTRAN STATEMENTS

This source language was designed as primarily a control logic for communication to and from the measuring machines. The various FORTRAN links that execute the reconstruction calculations are called by KERTRAN routines. We have tried to keep a parallel with the FORTRAN style as much as possible. The assumption is made that the reader is familiar with FORTRAN.

CODING FORMAT

The statement position format on an 80 column card is quite similar to the FORTRAN coding conventions. Columns 1-5 contain the statement label. Column 6 is reserved for a continuation character (any non-blank character is valid). Columns 7-72 may contain the KERTRAN statement.

COMMENT CARDS

Any card with a C in column 1 and a blank in column 2 is treated as a comment card.

STATEMENT LABELS

Unlike FORTRAN, the KERTRAN statement labels are not limited to numeric characters only. A label may consist of one to five alphameric characters. Its initial character must be alphabetic (not numeric). The following are valid labels:

```
ST100
A40B
BEGIN
```

The following are not valid labels:

```
100
A.60      (Special characters are not allowed)
AbLPH     (Blanks are not permitted in a label)
```

Labels need not necessarily begin in column one; however, they must not extend beyond column five.

VARIABLES

All variable names must consist of one to six alphanumeric characters. The first character must be alphabetic. Blanks are not allowed in a variable name.

All KERTRAN variables are treated as integer quantities. There exists no option in the language for the use of floating point variables. This does not turn out to be a severe restriction. KERTRAN has the ability to call a FORTRAN subroutine which could do necessary floating point calculations.

CONSTANTS

The only constants allowed in KERTRAN are integer constants not greater than 999 999.

VARIABLES

Variables may be non-subscripted or subscripted. The number of subscripts is limited to three. Computed subscript values are not allowed when reference is made to any array.

$A = B(I, NP)$ is valid, while

$A = B(I+1, J)$ is not valid.

ASSIGNMENT STATEMENTS

Arithmetic assignment statements are quite similar to those of FORTRAN.

$A(R, I) = B * C + E$

is a valid statement. Currently we have not invested a large amount of effort in the arithmetic statement compiler logic, so there exist the following restrictions with these statements:

1. The only operators allowed are *, /, +, -.
2. No parentheses are allowed.
3. Order of computation is a simple left to right process.

For example, the statement

$$A = A + B/C + D * E$$

would be evaluated as

$$A = \left(\frac{A+B}{C} + D \right) \times E.$$

Obviously this is one area of the compiler that we are going to revise as soon as possible.

IF STATEMENT

Transfer of program execution is accomplished by use of the statement

$$\text{IF}(V_1 \odot V_2) \text{LABEL}_1, \text{LABEL}_2$$

V_1 and V_2 are any valid KERTRAN variables. $\odot$ may be any one of the following logical operators:

EQ. EQUAL
LT LESS THAN
GT GREATER THAN

LABEL_1 and LABEL_2 are any valid statement labels. The logical expression within () is evaluated. If the result is true, control is transferred to the statement with the label LABEL_1 . If the result is false, control is transferred to the statement with the label LABEL_2 . It is not allowed to use any sign preceding V_1 or V_2 .

$\text{IF}(A.EQ.-10)\text{STP20,BEGIN}$

is not a valid statement. One would have to write something like

TEST = -10

$\text{IF}(A.FQ.TEST)\text{STP20,BEGIN}$

GO TO STATEMENT

Transfer of control may be accomplished by use of the unconditional GO TO.

GO TO LABEL₁

(LABEL₁ is any valid statement label). The computed GO TO is of the form

GO TO(LABEL₁, LABEL₂, ..., LABEL_n)V₁

LABEL₁, LABEL₂, ..., LABEL_n are statement labels and V₁ is a non-subscripted variable. The execution of this statement is identical to the FORTRAN computed GO TO.

DO STATEMENT

The form of KERTRAN DO statements is identical to those of FORTRAN.

DO LABEL₁ V₁ = V₂, V₃, V₄

V₁, V₂, V₃, and V₄ are non-subscripted variables. If the increment variable V₄ is omitted, an increment of one is assumed. LABEL₁ must be the label of a CONTINUE statement. Unlike FORTRAN, the CONTINUE statement may not be used for any other purpose. KERTRAN imposes no rules on the execution of DO statements. The programmer may, if he desires:

1. Start a DO with the start index greater than the end index.
2. Transfer control to the inside of a DO loop without executing the DO statement itself.
3. Change the value of the DO index, DO end variable, or the DO increment while in the loop.
4. Use the value of the DO index at any time.

Before one attempts to make use of these features he should be aware of the internal logic generated by a DO and CONTINUE. The logic for

DO ENDLP I = J,K,L is

A is an internal register not directly available to KERTRAN

Each DO must reference its own unique CONTINUE statement. In other words, no two DO statements are allowed to reference the same CONTINUE.

CALL STATEMENT

Subroutines written in FORTRAN or the EMR assembler language ASSIST may be called from a KERTRAN routine. The CALL statement is identical to the FORTRAN CALL. It is not allowed to use call parameters with this statement. All variables transferred to and from subroutines must be in COMMON storage.

All subroutines are core resident and are loaded into memory along with the KERTRAN routines and the KUBER measuring machine controller. (See pages 5 and 6).

When KUBER encounters a CALL statement that refers to a core resident subroutine, the interpretive mode of instruction execution is terminated and control is transferred directly to the subroutine. At this point, it is possible for the system to 'hang' if the subroutine has any serious program errors. The execution of the RETURN statement in the called subroutine permits KUBER to gain control once more. It is not allowed to call a KERTRAN routine from a FORTRAN or ASSIST subroutine.

The process of loading a reconstruction or kinematics link from drum storage to memory and execution of that link is via the statement

CALL LINK n

where n is the number of the Link. The current links which we are using and their respective functions are:

LINK #3	Processes fiducial measurements in a view
4	Processes a vertex measured in all views
5	Reads the next scan card

- 6 Processes a track in one view
- 7 Calculates Z coordinates of points on a track
measured in all views
- 8 Calculates reconstruction variables for a track
measured in all views
- 9 Writes tape record of event measurements and
reconstruction variables
- 10 Sets up variables for a kinematics fit to a desired
final state
- 11 Executes the kinematics fit
- 12 Tests a track for meeting stopping track requirements
- 13 A special GUTS debug link
- 14 A KUBER debug and utility link

Link No. 1 is executed when the entire system is loaded and is never executed again. Its function is to establish bubble chamber optics constants, etc. Link No. 2 is not used at this time.

When KUBER processes a CALL LINK statement, it will check to see if any LINK is currently loaded and not finished with its execution for some other job in the system. If so, the last LINK requested will wait until the previous one has completed its execution. This scheduling process does not cause any delay in the measuring process because a typical LINK will execute in approximately 1/6 sec.

INVOKE AND ENTRY STATEMENTS

The procedure used to 'call' a KERTRAN routine from another KERTRAN routine is the INVOKE statement. The format of this statement is

INVOKE NAME

where NAME is a label of a statement in some KERTRAN routine. Control is 'returned' to the calling routine when a 'RETURN' statement is executed. An INVOKE statement may also be used to execute a body of code contained in the same routine that issued the INVOKE. The following examples are all valid uses of the INVOKE statement:

```

C      KERTRAN MAIN

      EQUIVALENCE CLASS 1 SUM,1, ARRAY,2, NORD, 51, 1, 52

BEG    SUM = 0

      NORD = 10

      INVOKE EVEN

      INVOKE TOTAL

      INVOKE ODD

      GO TO BEG

TOTAL  DO LOOP 1 = 1, NORD

      SUM = SUM + ARRAY(1)

LOOP   CONTINUE

      RETURN

      END

C      EXAMPLE OF A KERTRAN SUBROUTINE

      ENTRY EVEN

      ENTRY ODD

      EQUIVALENCE CLASS 1 ARRAY,2, NORD, 51, BEGIN, 52, 1, 53

```

```

EVEN      BEGIN = 0
          INVOKE BUILD
RET        RETURN
ODD        BEGIN = 1
          INVOKE BUILD
          GO TO RET
BUILD      DO ENDLP I = 1, NORD
          ARRAY(I) = BEGIN
          BEGIN = BEGIN + 2
ENDLP      CONTINUE
          RETURN
          END

```

KERTRAN subroutines do not utilize a subroutine statement for name identification purposes. The ENTRY statement of the form

ENTRY NAME

is used in place of the subroutine statement. NAME must be a label in the KERTRAN routine. More than one ENTRY statement may be included in any routine.

The process used by KUBER for execution of the INVOKE and RETURN statements is the 'pushdown stack.' When an INVOKE is executed, the address of the next instruction is placed in the top of the stack and control is transferred to the code beginning with the label specified in the INVOKE. When a RETURN statement is executed, the address in the top of the stack is removed and control is transferred to this address.

MEASURING MACHINE COMMANDS

The communication and control commands used execute the following functions:

1. Light a specified view lamp.
2. Display a message to the measuring machine operator.
3. Display a numeric digit on a specified binaview display unit.
4. Read the current setting of the operator button board and the x,y encoder values.

LIGHT VIEW STATEMENT

The view lamp will be turned on when the statement

LIGHT VIEW V_1

is executed. V_1 is any valid subscripted or non-subscripted variable, or it may be an integer constant. Any view lamp that is on will be turned off.

DISPLAY MSSG STATEMENT

Each machine has a message display readout that consists of 36 operator messages. These messages inform the operator of the next measuring sequence expected and also of geometry failure conditions that require a remeasurement. The format of this statement is

DISPLAY MSSG V_1 .

V_1 is any valid subscripted or non-subscripted variable, or it may be an integer constant. The message corresponding to the value of V_1 will be displayed.

DISPLAY BINA STATEMENT

In addition to the message display readout, a series of BINAVIEW displays is used to show numeric information to the operator.

This information includes current frame number, grid location, track number, next point on the track being measured, etc. The statement format is

DISPLAY BINA V_1 ON V_2 .

V_1 and V_2 may be any valid subscripted or non-subscripted variables, or they may be integer constants. V_1 is the number to be displayed and V_2 is the Binaview unit to be used for the display. If V_1 is greater than 9, the BINAVIEW will be 'blanked' out. V_2 must not be larger than 15.

INDAT STATEMENT

The operator may communicate to the system via the 'button board' attached to each measuring machine.

Each column of the 'button board' contains eight relays that may be depressed by the operator. It is not possible to depress more than one relay in a given column at once.

When data is read from a machine, the setting of each column of the 'button board' and the x,y encoder values are input by KUBER. This data is kept in the KUBER registers denoted symbolically as

MMX --- x encoder setting

MMY --- y encoder setting

BB1 --- Button board column 1 setting

BB2 --- Button board column 2 setting

BB3 --- Button board column 3 setting

The KERTRAN statement that executes this read operation is simply
INDAT.

After this statement is executed, the KUBER data registers MMX, MMY, etc. may be referenced by the use of simple assignment statements.

Examples are

VERTX = MMX

VERTY = MMY

COLUM1 = BB1

IF (COLUM1. EQ. 8) REJECT, ST100

It is easy to confuse the KUBER data registers with valid KERTRAN variables. They are 'key words' of the language and may not be used like any other variable. The following statements are not valid.

VERTX = - MMX

VERTY = MMY + 10000

IF(BB1, EQ.8)ST20, ST30.

One must code instead of the above statements, the following

VERTX = MMX

VERTY = MMY

TEST = BB1

VERTX = - VERTX

VERTY = VERTY + 10000

IF(TEST. EQ.8)ST20, ST30.

When the KUBER data register values have been transferred to valid KERTRAN variables via the simple assignment statements, then the variables themselves may be used in other KERTRAN statements.

END OF TRACK CONDITION

The operator uses a button labeled 'END OF TRACK' to indicate that the next point she measures on a track is the last point. Ordinarily the routine that controls track measuring will expect a specified number of points to be measured on each track. If the track is too short for this number of points to be measured, the operator may use 'END OF TRACK' and measure fewer points. This condition may be tested using the statement

IF(ENDTRK) LESS, NOT

after the INDAT statement.

BAD DATA CONDITIONS

During the execution of an INDAT command several bad data conditions may occur. They are

1. Invalid datex code
2. I/O parity fail
3. Memory parity fail

When the x,y encoder settings are read into the computer memory, they are represented as a 'datex code.' KUBER converts this code to decimal. If, during the conversion process, KUBER detects an invalid 'datex code,' a logical register (referred to symbolically as BADDAT) is set 'true.'

Three words of data are actually read from the measuring machine. The button board settings are contained in one word and KUBER unpacks this word forming the symbolic registers BB1, BB2, and BB3. If during the read operation any one of these words causes an I/O parity fail or a memory parity fail, the logical register IOFAIL or MFAIL is set to 'true,' respectively.

These logical data condition registers may be tested by the use of the KERTRAN IF statement.

```

READ          INDAT
              IF(BADDAT)READ, RE1
RE1           IF(10FAIL)READ, RE2
RE2           IF(MFAIL)READ, RE3
RE3           X = MMX

```

This is an infinite loop to continue attempting a read operation until all BAD DATA indicators are false.

PROCEED STATEMENT

As a KERTRAN routine is being executed, the logic eventually reaches a point when the measuring machine operator must communicate with the routine. The communication we are referring to here is an indication from the operator that she has positioned her reticle to a desired location and is now ready for the routine to read the x,y encoder settings as a data point. An example of such a situation was given on page 10. Another form of communication occurs after a message is displayed to the operator. She must indicate to the system that she understands the message and is ready to start the next measuring sequence.

In both cases of operator communication the routine must 'wait' until she has given her response. The statement that executes this 'wait' is

PROCEED C₁, C₂.

C₁ and C₂ are acknowledge and enter interrupt controls. The acceptable key words for these controls are:

TASK --- Continue program execution.

IGNORE --- Ignore this interrupt and continue waiting.

Thus the statement

PROCEED TASK, IGNORE

would discontinue execution of the program until the operator pushed the acknowledge button. Execution would then continue with the next sequential statement. If the enter button was pushed, this action would be completely ignored by the system.

EQUIVALENCE STATEMENT

The method of declaring KERTRAN variables as being located in COMMON storage is much different than the FORTRAN convention. There are three classes of COMMON variables that may be used in a KERTRAN routine.

Class 0:

This class consists of all the variables that appear in the FORTRAN common statements. On page 12 we briefly described the table structure used in the controlling program KUBER. In general, this class is used to transfer information from KERTRAN routines to the FORTRAN links. For example, there are two arrays used in the links that contain the vertex measurements as IVERTX(I,J,K) and IVERTY(I,J,K) where I is the view, J is the vertex, and K is the measuring machine number. If these arrays have been defined as the third and fourth entries in the KUBER name table for class 0 variables, the EQUIVALENCE statement used in the KERTRAN routine that will use these arrays would be

EQUIVALENCE CLASS 0 IVERTX, 3, IVERTY, 4

Blanks may be used for commas in this statement. No dimension qualifiers

are necessary in the EQUIVALENCE statement. In fact, they are not allowed. KUBER will use the dimension information from the name table to calculate a memory address when these arrays are referred to.

Since the arrays in this example have the measuring machine number as the last subscript, this subscript should not be used in any KERTRAN statement using these arrays. For example, if a KERTRAN routine had previously executed an INDAT statement to read the encoder settings and the next logical step was to transfer the x,y settings of the encoders to the vertex arrays for view I vertex J, the KERTRAN statements would be

$$\text{IVERT}(I,J) = \text{MMX}$$

$$\text{IVERT}(I,J) = \text{MMY}.$$

When KUBER executes these statements, it will use the information in the name table for memory address computation. Since entries three and four would indicate that the arrays are to be machine indexable, KUBER will automatically supply the machine subscript.

It is not necessary to include in the EQUIVALENCE statement all of the variables in common. Only those used in the particular routine being written need appear. There is also no restriction on order of the list.

$$\text{IVERTY}, 4, \text{IVERTX}, 3$$

is just as valid as

$$\text{IVERTX}, 3, \text{IVERTY}, 4.$$

Classes 1 and 2:

These two classes have a more specialized usage than class 0. In general communication to and from reconstruction links consists of a number of control variables that may relay information such as current

view being measured, current track, frame number, reconstruction failure flag, etc.

All information read from the scan card such as roll number, frame, event topology, etc., is located in class 2. Class 1 is used primarily to transfer control information between the KERTRAN routines and the LINKS. Each of these variable classes may be considered as a vector 100 words long. The EQUIVALENCE statement assigns variable names to the specified elements of this vector. The statement

```
EQUIVALENCE CLASS 1 NVIEW, 12, NTRAK, 15
```

would be a valid means of defining the view number and track number variables in some track measuring routine which would call links for track processing.

Variables assigned to class 1 or 2 may be non-subscripted or singly subscripted. The statement

```
EQUIVALENCE CLASS 1 A, 26, B, 24
```

would allow one to treat B as a singly subscripted variable whose third element is the variable A.

All EQUIVALENCE statements must precede any KERTRAN statement using these variables. EQUIVALENCE statements allow continuation cards. A continuation card follows the FORTRAN convention of containing any character punched in column 6.

LOCAL VARIABLES

Any non-subscripted or singly subscripted variable that does not appear in an EQUIVALENCE statement is a 'LOCAL' variable. Storage for such variables is reserved in the KERTRAN routine being compiled. Storage

for singly subscripted variables is allocated by the DIMENSION statement. The format is identical to the FORTRAN convention. The use of double or triple subscripted variables in the DIMENSION statement is not allowed.

THE 'NEXT' FEATURE

In some instances it will be necessary for a KERTRAN statement to be executed immediately for the machine currently using the program without allowing the KUBER Job Selector to select another machine for execution. This is done by coding NEXT on the statement. The statements

```
I = I + 1      NEXT
      DISPLAY BINA I ON 5
```

would cause immediate execution of the binaview display statement after the variable I was incremented.

CONCLUSION

The reader who is interested in the KERTRAN compiler construction should refer to IS-2230.¹ A detailed description of the on-line control program, KUBER, is given in IS-2229.²

In general, the system approach we have used has been enthusiastically received by the physicists in charge of their particular experiments. It has also relieved the programming staff of many maintenance duties. No longer does it take a professional programmer to change an assembler language routine, but the physicist may alter his control program with very little effort.

We do not feel that the concept of a special purpose language is limited only to film measuring applications. Any process that involves

communication to and from some device that is controlled by a computer is a candidate for a special purpose language.

ACKNOWLEDGMENTS

The authors wish to thank Mrs. Rita Wagstaff and Miss Cheryl Cate for programming contributions to the system. We would also like to thank the physicists for their patience during the debug period of the system.

REFERENCES

1. C. L. Cate, R. L. Wagstaff, and W. J. Higby, "The KERTRAN Compiling System," IS-2230 (December 1969).
2. A. H. Klein and W. J. Higby, "KUBER: The Control Program for the USAEC Ames Laboratory On-Line Measuring System," IS-2229 (December 1969).
3. T. L. Schalk and W. J. Kernan, "Physics Use of the Ames Laboratory On-Line Measuring System for Bubble Chamber Events," IS-2231 (December 1969).

INDEX

INDEX TO KERTRAN STATEMENTS AND LANGUAGE FEATURES

	Page
ARITHMETIC ASSIGNMENT STATEMENTS	15
BAD DATA CONDITIONS	26
CALL LINK	19
CALL SUBROUTINE	19
CODING FORMAT	14
COMMENT CARDS	14
COMPUTED GO TO	17
CONSTANTS	15
CONTINUE	17
DISPLAY BINA	24
DISPLAY MSSG	24
DO	17
END OF TRACK	26
ENTRY	13
EQUIVALENCE CLASS	28-30
GO TO	17
IF	16
LIGHT VIEW	23
INDAT	24
INVOKE	21
KUBER DATA REGISTERS	25, 26
NEXT FRAME	31
PROCEED	27
STATEMENT LABELS	14
VARIABLES	15