

THE DEVELOPMENT OF A COMPUTER-
AUGMENTED PHOTOGRAPH MEASURING
SYSTEM

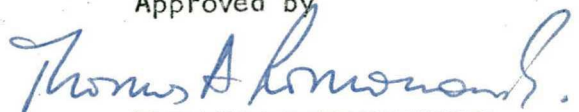
Presented in Partial Fulfillment of the Requirements
for the Degree Master of Science

by

John Walker Heimaster

The Ohio State University
1971

Approved by

A handwritten signature in blue ink, reading "Thomas A. Rounsford". The signature is written in a cursive style with a horizontal line underneath the name.

Adviser

Department of Physics

TABLE OF CONTENTS

Acknowledgment	ii
I. The Problem	1
II. The Hardware	8
III. The Software	12
IV. Summary and Results	25
Appendix I - Sample Output	28
Appendix II - Technical Notes	31
Appendix III - Software Summary	34
Appendix IV - Glossary	36
Appendix V - Memory Layout.	39
Appendix VI - Bibliography	40
Appendix VII - Flowcharts	41

ACKNOWLEDGMENT

I would like to express my gratitude to Dr. Thomas A. Romanowski, my adviser, for his guidance during this work. I wish to acknowledge the role of our engineers and technicians, Lloyd R. Alexander, Howard Dyke, Robert Salem, and Warren Shupp, who have designed, built, and maintained the equipment described here. I also wish to thank Mrs. Arleen Danford for her assistance with this manuscript, and Dr. Alan Stevens, Mrs. Betty Stevens, and our scanning staff for their patient suffering during the development of this system.

I. THE PROBLEM

1. Introductory remarks

In recent years, researchers in many areas have relied upon photography for the recording of experimental data. Photography has permitted rapid, precise data logging by automatic machinery. This has made possible experimental work at speeds too great for human comprehension, at light intensities too weak for human detection, and in environments too hostile for human presence. The ability to record data with high precision at a rate suited to the experiment, and then to analyze it at a rate suited to the experimenter, has now become an integral part of research practice in many disciplines.

The very attributes which make film a desirable data logging medium, primarily speed and compactness, have led to operational problems. Present research techniques often produce large volumes of film from a single experiment. The processing of this photographic data into meaningful scientific results involves great expense, and the maintenance of adequate records during this processing imposes a clerical burden on the experimenter. A variety of techniques have been developed to meet these problems; this thesis describes one such effort.

2. Background

At the time this work was undertaken, the High Energy Physics Laboratory at Ohio State was actively engaged in the analysis of a large experiment, which had yielded some 1.3 million stereoscopic pairs of

35 m.m. photographs. Each of these frames showed a series of particle tracks in spark chambers; each of these tracks appeared as a row of discrete sparks outlining the spiral path of a charged particle in a magnetic field. These tracks, photographed against a background of registry markings ("fiducials," provided for coordinate reference), constituted the primary data of our experiment.

The analysis of this film begins with a scanning operation, in which the frames are rapidly examined by a trained technician to eliminate those which obviously do not contain the particular particle interaction sought in our experiment; approximately 80% of our frames were eliminated at this point. The film is then scanned again, and detailed sketches are made of all likely frames as a guide for track measurement. These frames are manually measured on a "measuring machine," a high-precision film projector with a movable cross-hair; this device records the position of the cross-hair on the projected image directly into punched cards at operator-selected points.

The punched cards are now checked by various computer programs for human and mechanical errors in measurement; the measuring/checking cycle is repeated until acceptable measurements of each frame have been made. Magnetic tapes containing the accepted cards are then passed to elaborate analysis programs, which fit curves to the measured tracks; these programs can then deduce from these curves and the experimental parameters the identity and motion of the charged particles which have left the tracks. At this point the comparatively small fraction of frames containing the desired interaction can be located (about 0.1% in our

current experiment), and physics results extracted from the particle motions.

3. Difficulties

The application of the techniques described above to a million frames of film introduced a number of serious problems. The most obvious problem was clerical: extensive records had to be kept, tracing the processing of each frame. This work, tedious in the extreme, had to be performed with great care to prevent complete chaos. The remeasurement of rejected frames had a serious impact here; film had to be located and remounted, the data from remeasurements processed separately through all checks and inserted into the total data stream at its proper location, and all records had to be updated to reflect the remeasurement.

In order to insure the recording of all necessary data for each frame, a rather elaborate "measuring sequence" was followed in digitizing a frame; any deviation from this sequence resulted in the rejection of the frame by subsequent checking programs, perhaps a week or more after the error was made. This rejection for "sequencing" violations was a major cause of our remeasurements; the time lag involved was the major reason that remeasurement was burdensome. Moreover, this time lag made the training of new measurers quite difficult; the lack of immediate feedback made it impossible to accurately evaluate their work before large volumes of invalid data had been recorded.

The delay before validation of data introduced the additional risk of data loss by mechanical or electronic failure. The measuring machine, an elaborate electromechanical system, is susceptible to subtle failures.

Some of these invalidate all data without any detectable warning; others occur only occasionally and may frustrate investigation for long periods. Delay in error detection here can lose several days' work to a continuous failure, and may altogether prevent the location of an intermittent one.

4. A possible solution by further automation

A number of efforts have been made to solve the problems of film measurement by additional computer processing. In principle, the entire process of film analysis can be performed by a computer equipped with suitable film reading devices and programmed with sufficient ingenuity. In practice, the problems of pattern recognition by digital computer are still largely unsolved; existing systems require simple patterns and great amounts of time on substantial computers. For our purposes, this approach was impossible for both technical and financial reasons; however, we felt that the introduction of a small computer into our manual measuring system could enhance its reliability and speed, could automate our manual clerical process, and could provide rapid data checking. To achieve these ends, a small computer was purchased and development undertaken for both the necessary electronics and computer software.

5. Detailed goals for the measuring system

The goals for our measuring system were as follows:

- a) The system should be implemented on a small, inexpensive computer.

Our two measuring machines should be directly connected to this computer, along with additional devices for operator communication and

data handling. The devices developed for interconnection and communication should be convenient to use and reliable in service.

- b) The system should continuously monitor the measuring process on two independent measuring machines; it should immediately detect "sequencing" errors, guide the measurer through their correction, and output final data free of contamination by the now-corrected errors.
- c) The system should detect as many failures in the measuring machine system as possible, minimizing their impact on the data while informing the operator of the status of his measuring machine.
- d) The system should provide a typewritten log of film measurements; this should supplement manual records of film processing, and, by logging measuring machine failures, should assist hardware maintenance as well. A "machine-readable" record should be punched as well, as a first step toward totally automatic record keeping.
- e) The total reliability of the measuring system should be as high as possible, and should certainly exceed that of the electromechanical equipment it replaces. Moreover, by recording detailed failure information, the system should assist maintenance personnel in the isolation and correction of errors.
- f) Essential data compatibility should be maintained with the existing measuring system, allowing the use of that system for backup and minimizing the alterations required to our various data-handling programs.
- g) The system should be convenient and pleasant to run; clarity and simplicity of operation should be stressed in both hardware and software design. Fatigue and boredom should be reduced wherever possible.

h) A facility should be available to completely and rapidly check all data on our small computer. This facility, supplementing the continuous sequence checks made during measurement, should perform all validations previously made by our extensive checking programs operating on other computers.

i) Certain economies should result from this system:

Reduced clerical processing.

Reduced remeasurement, due to checks performed during measurement.

Simpler remeasurement, due to rapid error detection; remeasurements should be made before film and records are stored, before data is processed by subsequent programs.

Reduced data loss from systematic hardware and measure errors.

Reduced cost for externally-purchased computing.

In addition, the speed of the computer relative to the electromechanical measuring system should yield a higher basic measuring rate, as well as higher effective rate from simplified error detection and correction.

6. Foreground/background software

As the development of this system was undertaken, the need arose for general-purpose computing in the form of a simple, user-operated Fortran programming system to serve our researchers. It was felt that, with modest expenditures for additional hardware and programming development, our measuring control computer could serve both purposes simultaneously. The resulting system was to be of the form known as "foreground/background," in which the computer's facilities are shared between two

concurrent users - our measuring system, permanently held in the computer and given absolute priority for any computer time it requires (the "foreground"), and a series of general-purpose jobs supplied by our various users, given lower priority due to their lesser urgency (the "background"). A portion of the machine should be reserved for the foreground job (measuring control); this portion should be protected against any damage to its program by failures in background programs. The background should concentrate the remaining computer resources not required by the foreground toward the solution of user problems; the background user should be able to utilize these facilities without any knowledge of simultaneous foreground processing, and his errors should not in any way affect foreground operation.

To meet both these goals and those stated for our photo-measuring system, extensive hardware and software development was required. Equipment was designed, built, and tested which allowed direct control of the measuring process; additional equipment was developed to permit background processing. A substantial data-acquisition program was written to process measuring data in our foreground; numerous programs were written for background operation, including an elaborate data-checking program for measuring data. This thesis recounts these development efforts.

II. THE HARDWARE

1. The control computer

This system has been developed around a DDP-116 computer, manufactured by the Computer Control Corp. (now the Computer Control Division of Honeywell, Inc.). The machine is a typical small, general-purpose computer, possessing 12,288 16-bit words of magnetic core memory with a memory cycle time of 1.7 microseconds; the initial configuration included this machine, two Teletypes, and a magnetic tape unit. This computer, when obtained, had many of the deficiencies of similar small machines - limited instruction set, meager software, inadequate documentation - which we have, in part, remedied. The hardware required to do this, while neither designed nor constructed by the author, has been largely specified by him as a part of the total system design, and is therefore described here.

2. Computer modifications

A variety of hardware modifications to the computer were necessary to achieve the objectives of this project:

- a) A simple clock source was required to enable the computer to monitor events at fixed time intervals, and to maintain time-of-day. This clock time is derived from the power line as frequency source, using a small transformer, a gate, and a ripple counter; by initializing this counter to 4 and detecting overflow at 64, one second can be

easily timed. This one-hertz interrupt source and a modest amount of software provides the required capabilities.

- b) Special hardware was required to prevent untested background programs from halting computer operation or accidentally destroying the measuring control programs. Simple hardware was devised which constrains background programs to a fixed 8192-word area of storage and prevents computer halts; attempts to violate these constraints are detected and routed to recovery software before any damage occurs.
- c) No provision had been made by the computer designers for convenient initial loading of system programs into the machine following power-on or system collapse; the operator was required to enter a small program into the computer memory directly through the console switches. We have developed a circuit which transfers a program recorded on a bank of toggle switches into the computer memory and executes it, under control of a single button on the computer console. This design has permitted a simple, fixed operating procedure to be taught to our users, while complete flexibility is retained to alter the program recorded in the switches.
- d) A single push-button was added to the computer panel to permit the background user to terminate his program at will without manipulating the switches of the computer console. This technique, which operates completely through software, eliminates the danger of destroying the measuring programs or halting the computer by improper operation of the console.

Two peripheral devices were added to the DDP-116 to service background users. The first, a conventional card reader, provides program and data input; the second, a card punch, is available to both background and foreground users.

To allow this joint use of the card punch by two independent users, we have produced an interface of rather unusual design to operate the card punch. This interface is unique in its capability for autonomous error recovery - punch errors produce audible alarms via computer interrupt; the operator, summoned by the alarm is guided through a sequence of corrective actions until valid cards have been punched. By removing the error recovery function from the computer to the punch interface, many software difficulties are avoided; simple software can overlap data punching and punch error correction with data collection and background computing, without risk of data loss or operator confusion.

4. The data acquisition system

We have devised a simple, general data acquisition system which is largely independent of the detailed characteristics of our film and measuring process. Two interrupt sources are provided for each measuring machine, a foot pedal and a panel of push-buttons; two data inputs are accepted, an array of decimal thumbwheels and a group of data lines connected to the shaft position encoders of the measuring machine. (These may be interpreted as perfectly general signal sources, readily adaptable to other situations; however, in all subsequent discussions we shall interpret them as used in our measuring system.) The push buttons are

used to enter control commands (begin frame, erase point, etc.), supplemented by the thumbwheels for associated numeric inputs (frame number, measurer identification); the foot pedal requests the recording of point coordinates generated by shaft position encoders of the measuring machine. Each of these inputs has been buffered and interfaced to the measuring machine; each portion of the system has been adjusted for improved reliability, noise immunity, and ease of operation as our experience has increased.

5. The operator display

In addition to the measurer input devices described above, a measurer output device is required to display computer-generated error messages. A display box was accordingly constructed for each measuring machine, bearing two digital readouts ("Nixie tubes") and an audible alarm. A computer-generated error message appears as a two-digit error code and a continuous alarm; in addition, the alarm is sounded briefly after each accepted measurer action as a computer-generated acknowledgement.

This equipment, while providing the stimulus-and-response facilities required for smooth communication between computer and measurer, proved inadequate in service; it has been replaced by a display bearing five digits of additional, data-dependent status information. This additional data, continually revised as measurement progresses, assures the measurer that his actions are correctly interpreted; this has led to increased confidence and efficiency.

III. THE SOFTWARE

1. General introduction

The novelty of this system lies in its software. Hardware, a rigid medium, slow and expensive to modify, has been selected for simplicity, generality, and ease of construction; software, the more flexible medium, gives far greater freedom for experimentation to meet the diverse, often conflicting, goals described in the preceding chapters.

The concept of an operating system - a collection of programs allowing a computer to manage its own resources in accordance with predetermined goals - plays an essential role in modern computer system design. This software shapes the architecture of the machine as viewed by the programmer; the ease or difficulty of a programming task is as much determined by this software as by the physical capabilities of the hardware. Ten years' experience has substantially established desirable goals for large-machine systems; the major manufacturers are deeply committed to ever-increasing software capabilities at the expense of ever-increasing size and complexity of operating systems. Modern large-machine operating systems are characterized by elaborate schemes for sharing the resources of the computer system (processor time, storage, tape drives, etc.) among several concurrent users; utilizing some predetermined strategy, they attempt to achieve maximum overall efficiency for the several users. This goal, with the accompanying necessity to

13
protect each user from damage by the others while still providing him with virtually all the capabilities inherent in the machine, has led to systems of great complexity; unfortunately, these have themselves become substantial consumers of the computing resources which they allocate.

On small machines, the role of the operating system is less clearly established. The goals of the large-machine systems are still sound - resource sharing, error recovery, the execution of diverse tasks without human intervention - but the resources of the small machine, especially available storage, are so limited that each new operating system feature consumes critically needed facilities. Moreover, small machine instruction sets are so limited, and small machine input/output gear so diverse in nature, that large programs are often required to perform simple functions; these large programs are luxuries the small-machine user cannot afford. This conflict of highly desirable goals with harsh realities has led to a great diversity of small-machine systems, each attempting to utilize its machine to the utmost for some particular goal. Compactness has been achieved by clever coding, sacrifice of generality, and elaborate schemes for multiple utilization of a single storage area by program overlays.

2. Motivation for our software

Our design goals required the construction of a multiprogramming system - one which appears to perform more than one function simultaneously; the nature of our application permitted a limited version of such a system, the foreground/background configuration described in Chapter 1. This simplification of the system to only two concurrent users has had

far-reaching effects; in essence, it provides a strict priority for all¹⁴ resource allocation (to the foreground, whenever demanded) and a fixed order of control flow. In such a system, the foreground, aroused by an external signal, should process data so long as is required, then relinquish control to the background, which absorbs all available time until foreground processing is again required. Ideally, all foreground input/output operations should be overlapped in time with background processing; background input/output can require continual attention, since the background has no lower-priority user to absorb its unproductive wait time.

This arrangement demands software and hardware capable of responding to external signals, of switching control rapidly between two different functional tasks, and of resolving conflicts between the tasks for the computer's resources. The hardware necessary to achieve these goals in our environment has been described above - memory protection (to prevent the background user from inadvertently destroying the foreground user), external interrupt sources from the measuring apparatus (to demand service when required), and an elaborate card punch interface (to overlap foreground punching with background computation, to permit shared use of the punch by both foreground and background, and to provide error recovery with minimum software intervention). The software will be described in terms of independent logical units, each performing a specific group of functions; the nature of each unit will be discussed, as well as its role in the overall system.

3. Details of our software

- a) The monitor. This small piece of permanently-resident code provides a variety of essential services to both foreground and background users. All interrupts from external devices are directed to the monitor; some are completely processed by monitor code, while others must be directed elsewhere. A variety of background services are provided via a small "monitor call" subroutine, which presents a coded request from the background program to the monitor; in this manner, the user program, which cannot directly access the monitor program due to the memory protection, may obtain a number of useful services, ranging from fetching a program from the system tape to obtaining the date and time-of-day. In addition, facilities are provided to allow a foreground program to be entered and to assume executive control of processor operations; the software required to share the computer between two users is loaded with the foreground, and is thus present in the machine only when required, substantially reducing the size of the resident monitor.

A variety of interrupts is generated by various components of the system - memory protection, the clock, the card punch, etc.; each is initially processed by the monitor. On entry from the (single) interrupt linkage provided by the hardware, the memory protection is disabled and the machine status is saved to permit restoration of the interrupted process without damage; the interrupt is then analyzed and routed to an appropriate service program; upon return from this program, the machine status is restored and the user program

resumed. Memory protection violations, halts, and manual interruptions from the "cancel" button result in the termination of the background job; the scheduler is loaded to print an appropriate message and read the next job. Punch interrupts are used to maintain an internal record of the present "ownership" of the card punch, required for the assignment of the punch between foreground and background, and to detect any punch errors; if a foreground is present, punch interrupts are presented to it so that it may punch cards from its output queue, and so it may call the measurers' attention to any need for manual error correction at the punch. Clock interrupts are processed to maintain time-of-day. In addition, a special clock exit notes background activity and, after a fixed idle period (presently 30 seconds) schedules the running of hardware diagnostic programs in the background as the most profitable use of idle time; on each subsequent clock interrupt, the card reader is checked for the availability of job input, and, if any work is present, the diagnostic is terminated. All other interrupts are passed to the foreground, if present, or are processed as hardware failures.

A variety of services are provided to the background programs by the monitor. The user may, by presenting a coded request, obtain the loading of a program from the system tape (the Fortran compiler, the assembler, etc.), read and write a common parameter table shared by various system programs, terminate his job, request the assignment of the card punch, etc.; a special facility is provided here for linking a foreground to the monitor, allowing the foreground to be

brought into the system as a background job (from cards, for example), greatly simplifying the preparation and testing of foreground programs.

Special code is provided to initialize the monitor; it is not permanently resident, and is so placed in memory that it can be destroyed by any job needing its storage. This code is loaded from the system tape via the load circuitry described in the hardware chapter; it in turn reads the monitor into memory and obtains from the operator, via dialogue over the console typewriter, the data and current time-of-day; these form the basis for all internal timekeeping. The initialization code then signals end-of-job, allowing the monitor to obtain the "next" (i.e., first) job from the card reader.

- b) The scheduler. This transient program is brought into memory by the monitor whenever control cards are to be read. It reads these cards, prepares tabular data extracted from them, and requests (via monitor call) the loading of requested programs; it prepares a typed record of essential control cards (including processing times), and provides error checking and recovery.

Upon arrival, the scheduler determines from the parameter list passed by the monitor whether a job has failed; if so, an appropriate diagnostic message is prepared and typed (for example,

PROTECT VIOLATION AT OCTAL XXXXX,

where XXXXX contains the address of the offending instruction). If a job is terminating, a line of statistics is typed giving the run time of the job. In all instances, the next acceptable control card is found and processed. Parameters are extracted from the cards and tabulated for further processing; program call cards, such as Fortran,

result in a monitor call for program loading, while others (comment cards, pause requests, various parameter cards, etc.) are processed solely by this program. Processing continues until a program is requested or the input deck is depleted. The monitor may, depending upon the status of various internal parameters, fetch the scheduler back into memory after a processing program terminates (e.g., after a Fortran compilation); if no further control cards are required, however, the monitor will sequence through successive steps without reloading the scheduler; for example, the sequence

```

$JOB          *** EXAMPLE ***
$FORTRAN      GO
               .
               . (FORTAN SOURCE)
/*            .
               . (END OF FILE)
               .
               . (USER DATA)
/*            .
               . (END OF FILE)

```

compiles, loads, and executes a Fortran program without further scheduler loads after the \$FORTRAN card is read.

Thus uniformity of input processing simplifies a variety of programming functions. Any system program is available by direct call from the system tape; any system or user program may temporarily altered via standard control cards (for this run only). Thus even such specialized programs as our data collection foreground may be loaded and altered in a logical manner:

```
$JOB
$ABS      GO
          .
          . (absolute binary deck)
          .
/*
$PATCH   20134,000000
$JUMP     20000
```

will load the deck and alter a single location in memory after loading. From tape, the following performs the same function:

```
$JOB
$EXECUTE  FGND
$PATCH   20134,000000
$JUMP     20000
```

By elimination of any real distinction between processing programs and user programs, simplicity of operation is achieved, and the full protection and debugging facilities of the system are available to both.

c) The foreground. This large, transient program implements our data collection system; it is resident (occupying one-third of the machine) whenever we are operating measuring machines on-line. For ease of development and maintenance, it has been divided into two portions; one portion is largely independent of our data, providing common services such as typing and card punching, while the other is heavily data-dependent, verifying the sequence and success of manual operations.

The foreground is initiated as a background job; by a monitor request, it links itself to certain interrupt-processing exits in the monitor, activates the memory protect system, and, then terminates as a background job; this permits the scheduling of further background jobs, while the memory protection and direct monitor linkage secure

the foreground from physical damage and provide interrupt input.

Foreground interrupts are passed to a closed subroutine in the foreground by the resident monitor; non-measuring interrupts are processed entirely there - teletype interrupts cause the typing of the next character on the logging teletype, card punch "good card" interrupts cause the punching of the next card on the output queue, card punch "bad card" interrupts cause the sounding of alarms to obtain manual intervention. Measuring input interrupts (data inputs via foot pedal, command inputs via pushbuttons) require more extensive processing; for this, a special foreground job is started. This job switch is achieved by a simple core move; the monitor's interrupt save area is copied into the foreground and new contents are provided. When the monitor resumes the interrupted program after the foreground interrupt subroutine terminates, it will unwittingly perform a job switch. The various possible foreground functions are represented by positions in a table, the "commutator"; when the foreground job is started, this table is searched until an active entry is found and processed. Each time a function is completed, the table is searched again for further work; when no further work is found, the background's registers are again placed in the monitor's save area and an interrupt exit is taken - and the monitor unwittingly switches back to the background job at the point of interruption.

[Technical note: By this means, foreground interrupts are processed very quickly, in that they merely re-dispatch the computer. Foreground main-line code, running enabled for further interrupts, actually processes foreground data; this code may wait for resources if necessary,

merely suspending the background without disabling the interrupts from card punch or teletype, which will free storage by performing output. Safeguards are provided to prevent the acceptance of a second interrupt from any given source before the first from that source has been processed.]

Storage management has proved to be a most critical aspect of our foreground design. Of the 4096 words allocated to the resident monitor and the foreground, approximately 1400 are available for data buffers; in this area, input data queues are maintained for two measuring machines and output queues for the card punch and logging typewriter. Maximum efficiency has been obtained by subdivision of this area into a collection of chained buffers (i.e., data blocks of fixed size, each containing a pointer to the next block of its queue), which are used in common by all queues; with this design, the system may allocate storage to areas of high demand at the expense of functions not presently requiring storage. Common subroutines have been written to allow the processing sections of the foreground to conveniently obtain and release these buffers from the common pool, to maintain their private buffer queues, and to wait until output operations free additional storage in periods of buffer shortage; thus a buffer crisis leads to a uniform and recoverable performance degradation, rather than total collapse.

A number of device-dependent (but not data-dependent) subroutines are provided to operate our real-time devices. A card punch routine queues output for punching, and, driven by punch interrupts, activates

briefly at the completion of each card to sound alarms in case of failure or to prepare the next card for punching. A logging console driver receives messages to be typed, appends date and time-of-day to each, and queues them to be typed when the typewriter becomes available; in addition, it permits the entry of a few control commands on the typewriter keyboard. Special programs read our measuring machines and convert their involved code to binary, while others read and decode our push-button panel for measurer input of commands; each of these programs contains extensive error checks and provides information (via the typewriter console log) on any hardware failures detected, thereby aiding maintenance of intermittent failures. Additional programs operate our error display and measuring machine lamps; various miscellaneous programs (date and time-of-day conversion, binary to decimal conversion, double-precision integer arithmetic, etc.), providing services to the entire foreground code, constitute the remainder of the data-independent foreground code.

The remaining foreground code processes data input by our measurers. To do this it contains detailed knowledge of film format and measuring sequence; this is the only foreground code explicitly dependent on these details. For the purposes of this code, each measuring machine status is completely described by a single data block. In it are recorded various internal pointers (input buffer queue, storage area for next data point, etc.) plus a complete status description of the measuring operation currently in progress. The measuring machine is considered to be operating in one of four modes - inactive (between frames of

film), measuring "principal fiducials" (i.e., registration marks, of which there are a fixed number in a film frame), measuring optional calibration fiducials, or measuring particle tracks; measurer input is interpreted in the context of the current mode. The foreground program, initiated after an interrupt, obtains the status block for the interrupting machine; the input is checked for certain possible hardware failures, then for logical consistency (e.g., point erasure requires recorded points). If the data is accepted logically, it is recorded in the status block or the measuring input queue and the measurer is notified of acceptance. If the data is not accepted, the measurer receives a coded error message on his display box and the input is not recorded. In either case the status display is updated to reflect the revised state of the measuring operation on this machine. When the measurer has recorded one complete "view" (one-half of a film frame), his accumulated data is transformed into card images and queued for punching; at the end of a frame, logging records are printed and punched.

d) Data checking and validation. To achieve the desired reliability and data accuracy of this measuring system, more extensive data checking is required than is performed by the foreground programs; this need is filled by our background checking system, which checks one batch of data as the next is being recorded. The necessary programs were written in Fortran by two members of our research group (Dr. Alan Stevens and Mr. Mark Hopkins), utilizing programming facilities prepared by the author; these programs provide immediate, detailed validation of the data punched by the foreground, noting mis-identification of reference points (fiducials), inaccurate measurements (excessive deviations),

and other errors too intricate to be readily detected by our foreground within its memory and time constraints. Regularly scheduled checking runs made throughout the working day provide routine error detection and allow scheduling of correction; additional runs, made whenever desired, allow rapid location of hardware faults, simplify training of new measurers, and allow rapid investigation of any "suspicious" circumstances reported by any measurer or technician.

- e) Background software support. In addition to the operating and measuring system described above, much service software has been obtained or written. The manufacturer's Fortran compiler, Dap assembler, Fortran library, and loader were extensively modified to fit out operating environment and needs; his input/output programs were altered, and others were developed to support our locally-constructed devices. Utility programs were written to build and modify system tapes, to handle a variety of convenient machine language formats on tape and cards, and to perform various conversion and copy operations between devices; scientific routines were obtained (e.g., the IBM Scientific Subroutine Package) or developed; hardware diagnostic programs were prepared for our locally-constructed devices. The programs, together with the special programs described in detail above, comprise our software system.

IV. SUMMARY AND RESULTS

The measuring system has been in operation for about six months. During this time it has been substantially revised, in both hardware and software, in the light of our operating experience; moreover, we have learned much about human behavior in a computer-regulated system. Despite the many problems which have plagued our system during its development, we are now realizing substantial savings in operating cost and complexity; we are beginning to show improvements in equipment maintenance and clerical support as well. Each of these aspects of our results will be detailed below:

1. Operational results

Our electromechanical measuring system, which this system replaces, has averaged four frames of film measured per hour of operation per measuring machine. These frames receive only the most primitive checks during their measurement; detailed checks are made by a series of computer programs on various machines, with a time delay of approximately ten days before a given frame has been fully validated by all programs. Thus, the response time to human error is very long, too long to be useful in training of new operators. The response time to measuring machine failure is intolerably long, permitting the loss of many days' work to mechanical failures.

Our computer-based system is now averaging six to eight frames of film measured per hour per measuring machine, a 50-100% increase in output.

The data is validated extensively during measurement, and is routinely checked several times a day by our background checking program; any dubious measurement or suspicious circumstance may be individually checked in a matter of minutes. Full validation is achieved within hours routinely, and minutes if desired. In this way, the volume of data within the "checking cycle" at any instant is kept manageably small and the resulting bookkeeping is minimized; also, failing frames may be remeasured before the roll of film has been dismounted and stored. The quick detection of errors improves our personnel training effort and minimizes the impact of hardware failures on our data production.

2. Operational problems

Our difficulties have fallen into two broad categories - the purely technical problems of hardware and software, and the human problems associated with computer-regulated control systems. The technical problems have primarily concerned hardware reliability and reproducibility. Specifically, each piece of our equipment has required individual effort to provide acceptable mechanical stability and noise immunity.

Our problems with human operation of the measuring system have proven more difficult. Measurers have displayed exceptional flexibility and patience as this system has evolved; they have been faced with real ambiguities of computer response and recording. Some of these ambiguities have been resolved by hardware modification - the foot pedals, pushbuttons, and associated electronics have been extensively reworked for increased reliability and certainty of operation. Others have been solved by software checks and still others vanished with the improved operator display, which provides detailed visual status information; every manual

operation now produces a noticeable change in the displayed value, greatly reducing measurer uncertainty. Each refinement has led to increased measurer confidence, with associated increased productivity. We are continuing to experiment with new types of manual controls in an attempt to select equipment most suitable across our range of measurers, and with various hardware and software delays, in an attempt to adopt the system to their natural work rhythms.

3. Conclusions

This system is now in operation on two measuring machines, 14-16 hours per day, seven days per week. An immediate increase in productivity of approximately 50% has resulted, and more may be expected as we grow more familiar with the equipment and refine our operating procedures. Our foreground/background configuration has made 95% of the computing time of the DDP-116 available for checking and general computation; these facilities will allow further improvements in checking, automated book-keeping, and data processing, as well as general program development for our research group. Background facilities are also utilized for diagnosis and testing of both the computer and its peripherals, thus contributing to the overall reliability of the system.

APPENDIX I - SAMPLE OUTPUT

The following pages show sample output from our measuring log and our data checking program. Each entry on the log is of the form

	MM/DD/YY		HH:MM:SS		MX -- message,
where	MM/DD/YY	=			date
	HH:MM:SS	=			time
	X	=			measuring machine number;

note especially the error messages for intermittent hardware failures at 18:36:04, 18:58:23, and 19:13:01. The checking run shows the processing of 13 frames, 3 of them containing errors; run time for the checking was approximately 6 1/2 minutes.

05/17/71 17:17:50 M2 -- RUN 1695, FRAME 10839886 STARTED BY MEAS. 10
 05/17/71 17:26:49 M2 -- VIEW 1 ENDED.
 05/17/71 17:40:42 M2 -- VIEW 2 ENDED.
 05/17/71 17:40:43 M2 -- FRAME 10839886 ENDED.
 05/17/71 17:41:10 M2 -- RUN 1695, FRAME 10839886 STARTED BY MEAS. 10
 05/17/71 17:46:00 M2 -- VIEW 1 ENDED.
 05/17/71 17:50:09 M2 -- VIEW 2 ENDED.
 05/17/71 17:50:10 M2 -- FRAME 10839886 ENDED.
 05/17/71 17:52:06 M2 -- RUN 1695, FRAME 10839886 STARTED BY MEAS. 10
 05/17/71 17:54:25 M2 -- VIEW 1 ENDED.
 05/17/71 18:02:33 M2 -- VIEW 2 ENDED.
 05/17/71 18:02:34 M2 -- FRAME 10839886 ENDED.
 05/17/71 18:03:42 M2 -- RUN 2695, FRAME 10839886 STARTED BY MEAS. 10
 05/17/71 18:05:33 M2 -- VIEW 1 ENDED.
 05/17/71 18:13:41 M2 -- VIEW 2 ENDED.
 05/17/71 18:13:41 M2 -- FRAME 10839886 ENDED.
 05/17/71 18:14:53 M2 -- RUN 0695, FRAME 10839899 STARTED BY MEAS. 10
 05/17/71 18:17:30 M2 -- VIEW 1 ENDED.
 05/17/71 18:22:30 M2 -- VIEW 2 ENDED.
 05/17/71 18:22:30 M2 -- FRAME 10839899 ENDED.
 05/17/71 18:22:46 M2 -- RUN 0695, FRAME 10839899 STARTED BY MEAS. 10
 05/17/71 18:25:30 M2 -- VIEW 1 ENDED.
 05/17/71 18:29:15 M2 -- VIEW 2 ENDED.
 05/17/71 18:29:15 M2 -- FRAME 10839899 ENDED.
 05/17/71 18:31:44 M2 -- RUN 1695, FRAME 10839962 STARTED BY MEAS. 10
 05/17/71 18:34:21 M2 -- VIEW 1 ENDED.
 05/17/71 18:36:04 M1 -- NO BITS SET ON BOX.
 05/17/71 18:41:48 M2 -- VIEW 2 ENDED.
 05/17/71 18:41:48 M2 -- FRAME 10839962 ENDED.
 05/17/71 18:42:39 M2 -- RUN 1695, FRAME 10839962 STARTED BY MEAS. 10
 05/17/71 18:44:41 M2 -- VIEW 1 ENDED.
 05/17/71 18:50:50 M2 -- VIEW 2 ENDED.
 05/17/71 18:50:51 M2 -- FRAME 10839962 ENDED.
 05/17/71 18:55:24 M2 -- RUN 2695, FRAME 10839962 STARTED BY MEAS. 10
 05/17/71 18:58:06 M2 -- VIEW 1 ENDED.
 05/17/71 18:58:23 M1 -- NO BITS SET ON BOX.
 05/17/71 19:03:10 M2 -- VIEW 2 ENDED.
 05/17/71 19:03:11 M2 -- FRAME 10839962 ENDED.
 05/17/71 19:03:15 M2 -- RUN 2695, FRAME 10839962 STARTED BY MEAS. 10
 05/17/71 19:05:29 M2 -- VIEW 1 ENDED.
 05/17/71 19:11:49 M2 -- VIEW 2 ENDED.
 05/17/71 19:11:49 M2 -- FRAME 10839962 ENDED.
 05/17/71 19:13:01 M2 -- MULTIPLE BITS SET ON BOX.
 05/17/71 19:13:12 M2 -- RUN 0695, FRAME 10839981 STARTED BY MEAS. 10
 05/17/71 19:15:51 M2 -- VIEW 1 ENDED.
 05/17/71 19:20:45 M2 -- VIEW 2 ENDED.
 05/17/71 19:20:45 M2 -- FRAME 10839981 ENDED.
 05/17/71 19:21:34 M2 -- RUN 0695, FRAME 10839981 STARTED BY MEAS. 10
 05/17/71 19:23:42 M2 -- VIEW 1 ENDED.
 05/17/71 19:31:14 M2 -- VIEW 2 ENDED.
 05/17/71 19:31:15 M2 -- FRAME 10839981 ENDED.
 05/17/71 19:33:51 M2 -- RUN 0695, FRAME 10839994 STARTED BY MEAS. 10
 05/17/71 19:35:54 M2 -- VIEW 1 ENDED.
 05/17/71 19:39:31 M2 -- VIEW 2 ENDED.
 05/17/71 19:39:31 M2 -- FRAME 10839994 ENDED.
 05/17/71 19:39:45 M2 -- RUN 0695, FRAME 10840015 STARTED BY MEAS. 10
 05/17/71 19:41:42 M2 -- VIEW 1 ENDED.
 05/17/71 19:48:02 M2 -- VIEW 2 ENDED.
 05/17/71 19:48:03 M2 -- FRAME 10840015 ENDED.
 05/17/71 19:48:49 M2 -- RUN 0695, FRAME 10840015 STARTED BY MEAS. 10
 05/17/71 19:50:24 M2 -- VIEW 1 ENDED.

miss I
5/14/71
4:00 P.M.

Pg 6

15:03:53 \$JOB
FRAME 416982. GOOD

CHECKING PROGRAM

FRAME 416986. GOOD

FRAME 417001. GOOD

FRAME 417015. GOOD

FRAME 417046. GOOD

FRAME 417057. GOOD

FRAME 417057. GOOD

THE X-OPTION HAS 4 FRONT FIDUCIALS AND 5 REAR FIDUCIALS IN CHAMBER 13.
THE X-OPTION HAS 4 FRONT FIDUCIALS AND 3 REAR FIDUCIALS IN CHAMBER 16.
BAD X-OPTION IN FRAME 417014. VIEW 2

THERE ARE 0 FRONT FIDUCIALS AND 1 REAR FIDUCIALS IN CHAMBER 4.
THERE ARE 3 FRONT FIDUCIALS AND 2 REAR FIDUCIALS IN CHAMBER 5.
FRAME 417072. @HAS 2 ERRORS IN VIEW 1

FRAME 417081. GOOD

THERE ARE 1 FRONT FIDUCIALS AND 0 REAR FIDUCIALS IN CHAMBER 8.
FRAME 417090. HAS 1 ERRORS IN VIEW 1

THERE ARE 0 FRONT FIDUCIALS AND 1 REAR FIDUCIALS IN CHAMBER 1.
THERE ARE 1 FRONT FIDUCIALS AND 0 REAR FIDUCIALS IN CHAMBER 3.
FRAME 417090. HAS 2 ERRORS IN VIEW 2

FRAME 417092. GOOD

FRAME 417096. GOOD

SUMMARY OF FRAMES

GOOD	10
BAD	3
DELETED	0

TOTAL	13
-------	----

LIST OF BAD FRAMES

417014.
417072.
417090.

STOP AT OCTAL 01620

15:10:29 EOJ -- ELAPSED TIME 00:06:36.

D.000010	P.000000	L.17040
D.000010	P.000000	L.17040

APPENDIX II - TECHNICAL NOTES

A number of features of the software developed for this project are of sufficient generality to merit some detailed technical comment; the discussion below, directed toward the experienced programmer, is not essential to the understanding of the remaining text of this thesis, and demands some knowledge of current software terminology.

1. The dispatching technique used to switch between foreground and background is unique, so far as the author has been able to ascertain. The monitor enters the foreground at a closed subroutine whenever an unrecognized interrupt occurs; the status of the machine at the time of the interrupt is recorded in the monitor's save area, whose address is known to the foreground. This asynchronous subroutine, operating disabled, classifies the interrupt and branches to the appropriate service routine. Some of these routines can completely process their interrupt at this time; the punch, for example, may be loaded with next card immediately. Other interrupts require processing by synchronous code, and must therefore effect a dispatch of the foreground program; a measurer pushbutton interrupt, for example, may require the acquisition of data buffers at the risk of a wait, an intolerable risk for disabled code. A redispach, if required, is achieved by the simple expedient of swapping the contents of the monitor's save area with those saved

for the task to be dispatched. A normal exit is now taken from the closed subroutine and the monitor restores machine status from its save area to resume normal processing; if the contents of this save area have been swapped, a redispach results.

2. Subtask selection within the foreground is accomplished via a simple commutator of the form

FGND	INH		disable interrupts.
X1	SKP		
	JMP	X1A	service routine #1
X2	SKP		
	JMP	X2A	service routine #2
	.		
	.		
	.		
	JMP	IDLE	exit to background

The interrupt service subroutine opens the gate corresponding to his synchronous service routine before dispatching the foreground; to do this, the skip (SKP) instruction is changed to a null (NOP) instruction, allowing the branch (JMP) following to be taken. When the foreground is dispatched, control passes to location FGND; the skips are successively executed until a NOP is encountered, and the appropriate service routine is activated. The routines, operating enabled, may wait for resources if necessary.

When the service routine terminates, it branches to FGND. If any other requests are pending, they are serviced in the order of the chain; eventually, all requests will be completed, and control will pass to IDLE to dispatch the background job.

3. Storage management for the foreground, which must operate with 1.5K of buffers in 3.5K total storage (16 bit words), is particularly demanding. For maximum efficiency, all available foreground storage is divided into fixed-length buffers; buffers from this pool are used for input data, output cards, and logging typewriter output as needed. Buffers are extracted from this pool by synchronous code, to receive measurer input or data output. They are added to the user's private queue; they are removed from this queue by external interrupts (for punch and typewriter buffers), or by measurer action (for input data). When no longer needed, they are returned to the common pool for reuse. At the end of a view on a measuring machine, 5-10 input buffers containing binary data are processed to yield 25-40 card buffers to be punched and two lines of log to be typed, while 5-10 other buffers containing data from the other measuring machine must be preserved; with less than 40 total buffers available, the resulting delay seldom exceeds five seconds.

4. With over 99% of CPU time available for background processing, the background is idle much of the time. To make most efficient use of this time, the system occupies this idle time in running memory diagnostics. The loading and execution of the diagnostic is scheduled by a special clock exit. When the background is not processing a job, the exit monitors the idle time; when thirty seconds have elapsed without background activity, a request for the loading and execution of the diagnostic is simulated. As the diagnostic runs, the card reader is tested at each clock interrupt (once per second); when the reader is readied, indicating the presence of a job, the diagnostic is cancelled to schedule the next job.

APPENDIX III - SOFTWARE SUMMARY

The following software was developed for this system:

MONITOR - the resident monitor for the system containing user service and interrupt code.

SCHEDULER - the control card analyzer for the systems.

FOREGRND - the photo-measuring system and the foreground/background code.

OS-EDIT - the system tape writer.

SERVICE - a collection of small utility programs which transfer binary files.

LDR-PCH - a program to read and punch absolute (core-image) binary decks.

UTILITY - a program to transfer data among various peripheral devices, with simple reformatting capabilities.

I\$CA, I\$CB, C\$CARD - programs to operate the card reader.

O\$CA, O\$CB, C\$PUNC - programs to operate the card punch.

BREAD, BPUNC - blocked binary card routines.

DLDA, etc. - double precision integer arithmetic.

TYPE - console typewriter driver.

BLOCK/DBLOCK - programs to process compact magnetic tape formats.

F\$TAPE, RDTAPE - programs for Fortran magnetic tape operations.

F\$ER, ERRSET, EXIT - Fortran error/stop service routines.

F\$W1, F\$R5, F\$W6, F\$W7, F\$R8, F\$W8 - improved Fortran I/O drivers.

The following distributed software was modified to suit our operating environment:

FORTRAN⁴, F4-10S - Fortran compiler.

DAP-16, 10S-16B - Dap assembler.

LDR-APM - relocating loader.

ASR-DUMP - core dump program.

In addition, numerous Fortran library subroutines were altered for accuracy or efficiency, and diagnostic programs were developed for our special hardware.

APPENDIX IV - GLOSSARY

To clarify the technical terminology used in the text, the following partial glossary is included:

background - an area reserved for general-purpose computation, to be performed at low priority during time not required by more urgent programs.

control system - a collection of equipment which regulates a process according to a predetermined strategy.

foreground - an area reserved for special-purpose computation, performed at high priority in response to demand; normally this code controls an external process regulated by the computer.

foreground/background system - computer software which enables the concurrent operation of a background program and one (or more) foreground programs; priority is given to foreground processing when required, with background processing during foreground idle time.

interrupt - a mechanism whereby an external device requiring computer attention may request the computer to suspend its normal processing sequence to service that device. The associated computer hardware suspends the current process, saves the status of the computer, and executes pre-set "interrupt code"; when this code terminates software normally resumes the suspended process.

memory protection - computer hardware which prevents user programs

from destroying memory belonging to the system or other users.

monitor - software which maintains actual control of a computer, during

the execution of user jobs; this code oversees the user

job, detecting and recovering a variety of user errors, scheduling

input/output operations, and performing a variety of services

for the user.

multiprogramming - the sharing of a computer among several independent

programs, giving the appearance of simultaneous processing

of the separate programs by rapidly switching the computer

among them.

on-line - direct computer operation of a device or process during its

occurrence, with response sufficiently rapid to guide its behavior.

operating system - a collection of software which enables a computer

to allocate its own resources among conflicting demands using

some pre-determined strategy.

overlay - a mechanism whereby a section of computer memory is used for

a variety of purposes successively, with contents supplied from

some external device; in this way, programs larger than the memory

capacity of the computer may be run in segments small enough

to fit.

peripheral - an external device controlled by a computer, such as a

printer or tape drive.

real-time control - regulation of a process on the time scale of that

process; that is, all necessary computations must be performed as fast as the process in order to regulate its flow.

resident - that portion of a program which is contained in the computer memory throughout the entire execution of the program; opposed to "transient."

resource - any of the facilities of a computer system available for allocation by the operating system, such as memory, tape driver, or computing time.

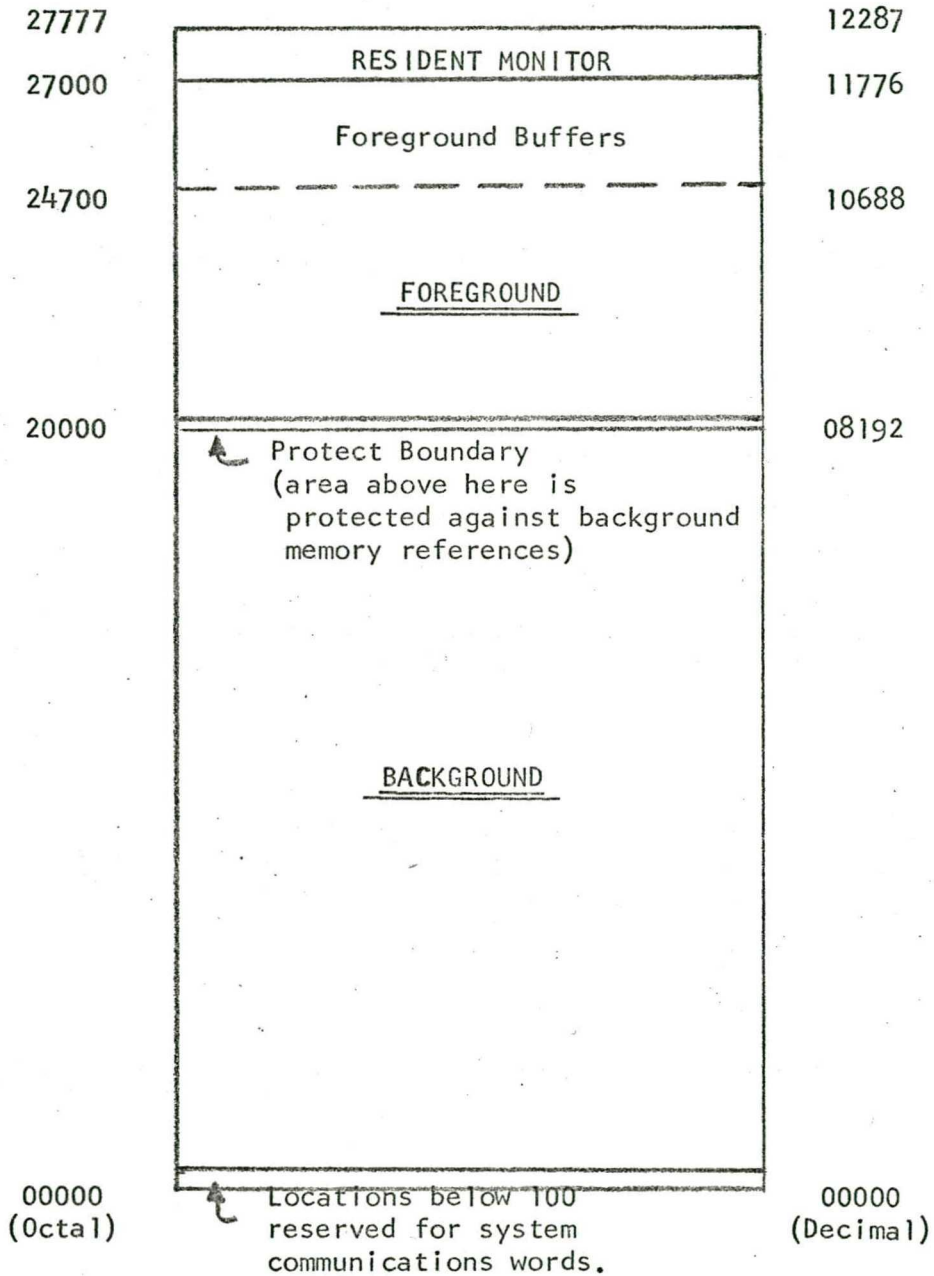
scheduler - that portion of an operating system which interprets user control input and selects user jobs for processing on the basis of that input.

task - an individual unit of work to be performed by a computer.

- - - - -

dispatch - to discontinue computer processing of one function and initiate processing of another under monitor control. Normally, this describes a software decision to switch between jobs in a multiprogramming system.

APPENDIX V -- MEMORY LAYOUT



This diagram shows the location of the various components of our system in the memory of the DDP-116 computer.

APPENDIX VI - BIBLIOGRAPHY

For a general introduction to computer-regulated control systems:

J. MARTIN, Programming Real-Time Computer Systems, Prentice-Hall,
New York (1965).

For a description of the DDP-116 computer:

Programmers Reference Manual for the DDP-116 General Purpose Computer,
Computer Control Corporation, Framingham, Mass. (1966)

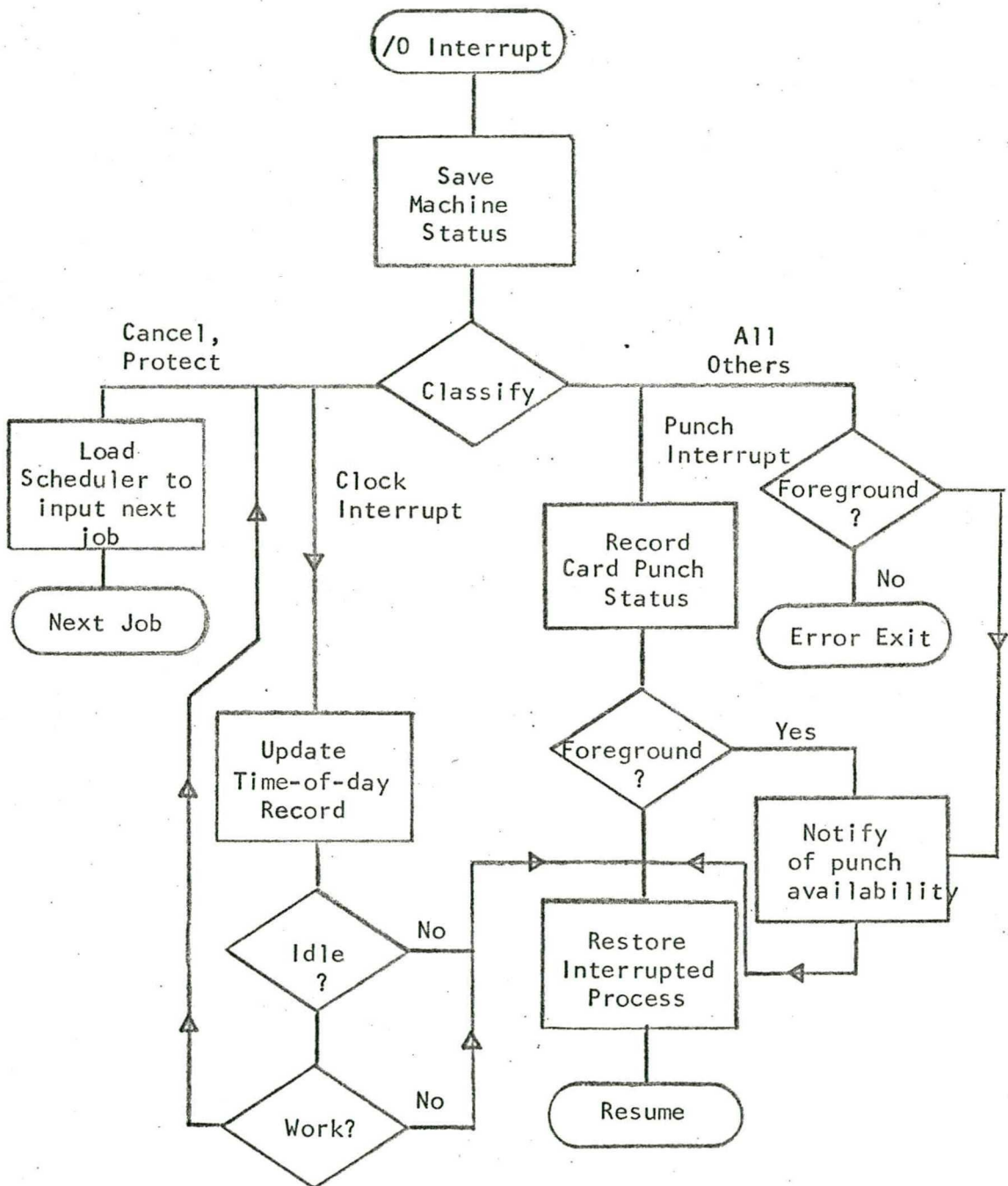
For a description of systems which have influenced the design of
this system:

1. IBM 7090 - 7040 Direct Couple Operating System Systems Programmer's Guide, IBM form C28-6382.
2. IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide, IBM form C26-3717.
3. Data Acquisition Multiprogramming System (DAMPS), Version 2, Application Description Manual, IBM form H20-0494.
4. The Hasp System, Version 3.0, IBM form 360D-05.1.014.
5. SDS Sigma 5/7 Basic Control Monitor, SDS form 90-09-53B.

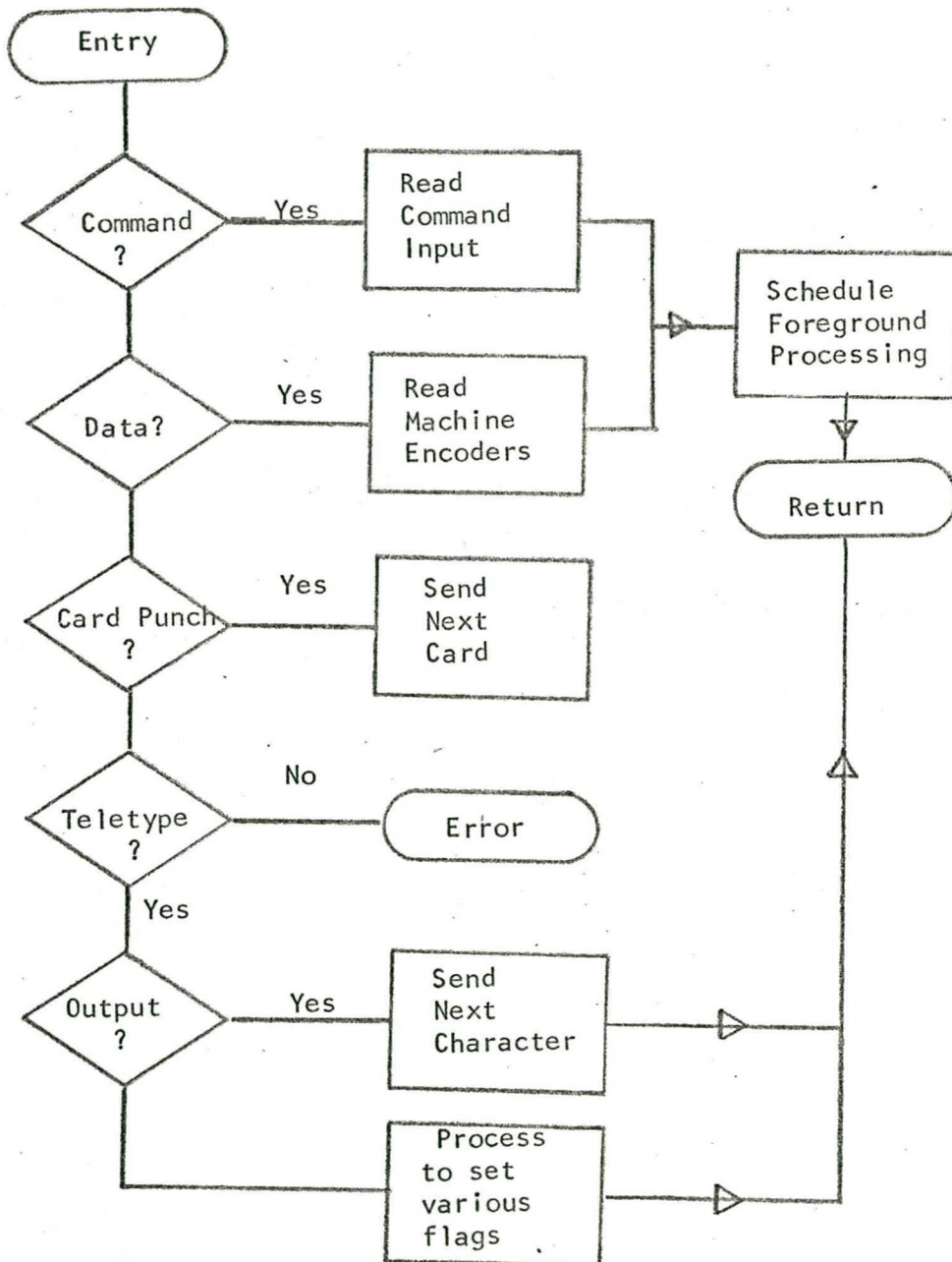
APPENDIX VII -- FLOWCHARTS

The following pages contain flowcharts of several important routines in the resident monitor and the foreground, indicating typical paths for processing our real-time data.

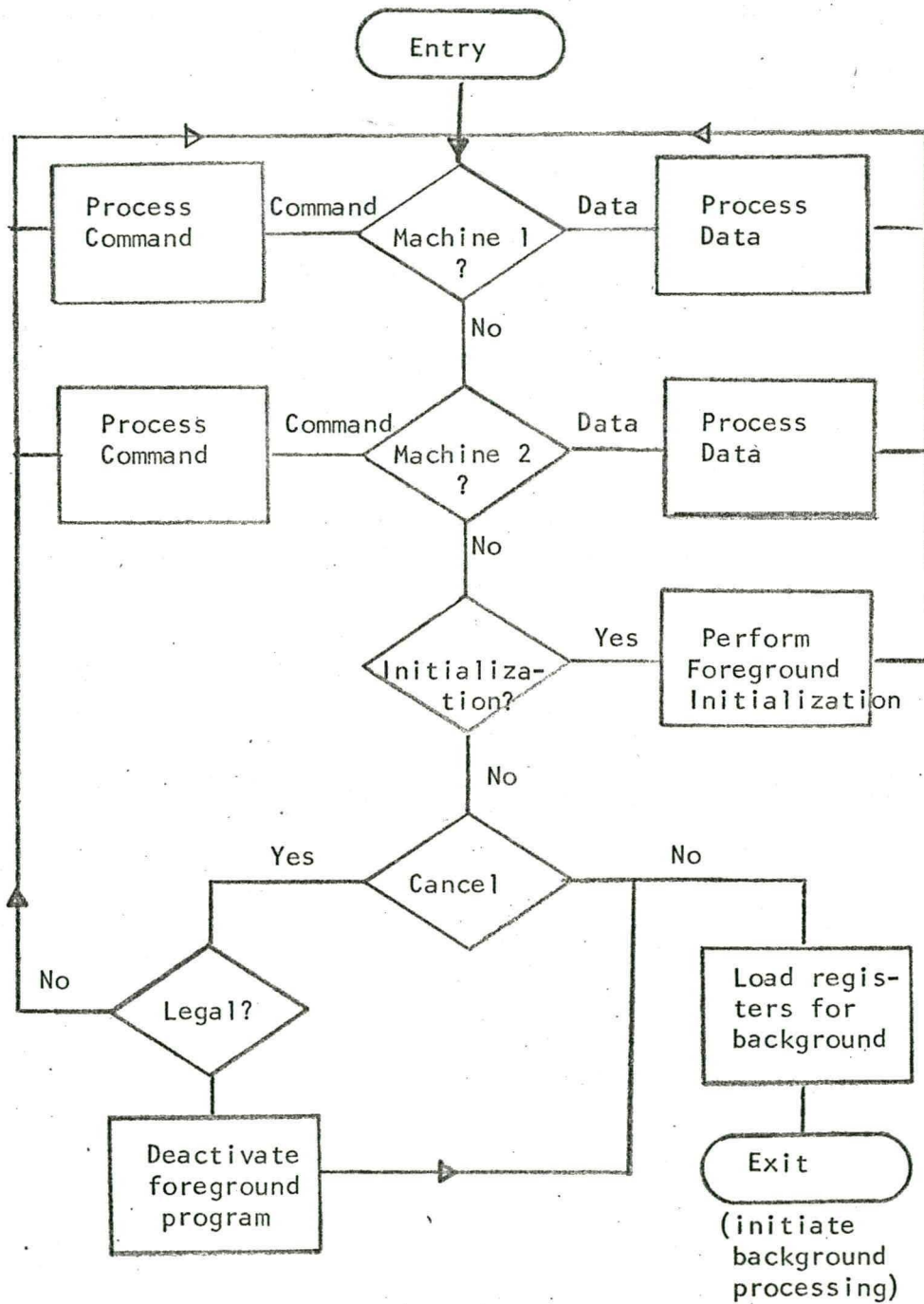
MONITOR INTERRUPT SERVICE



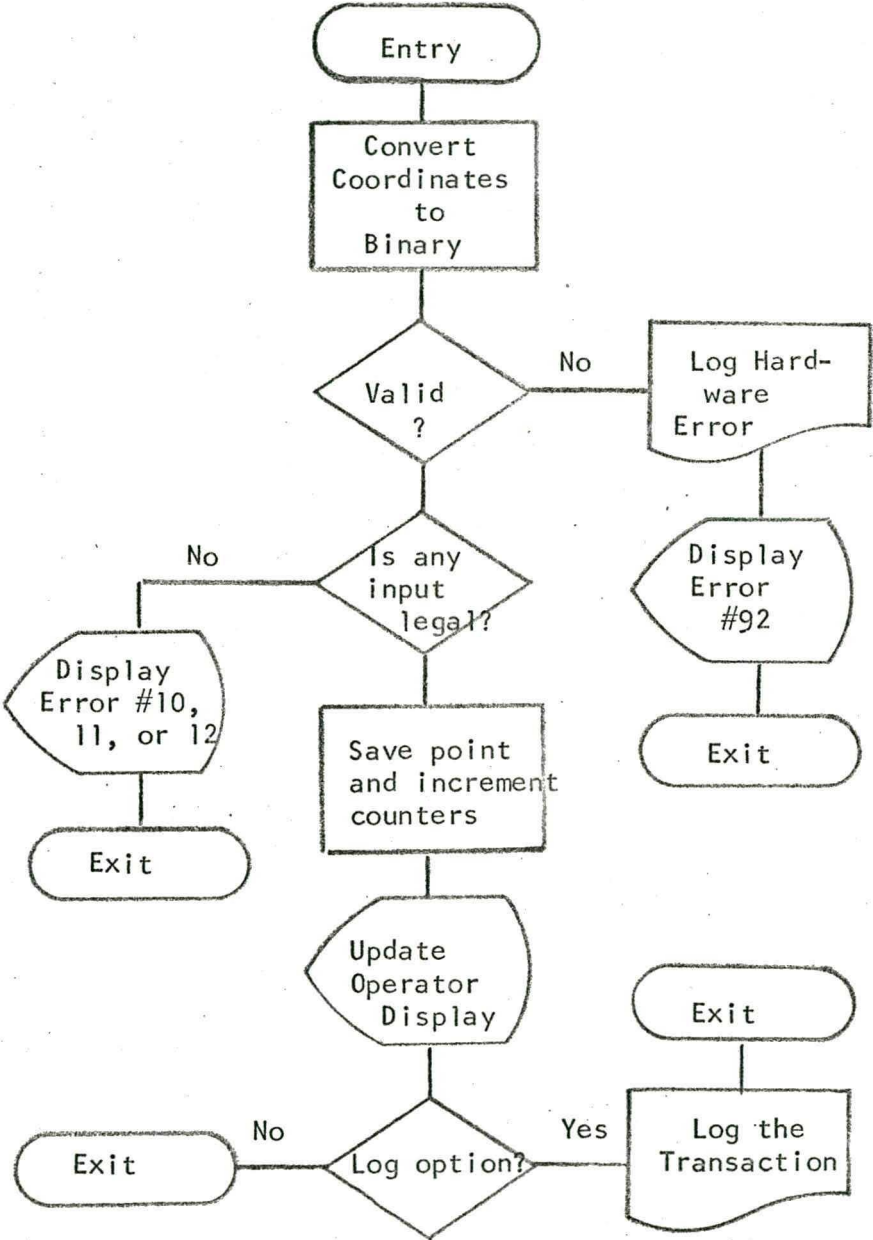
FOREGROUND INTERRUPT SERVICE



FOREGROUND PROCESSING



FOREGROUND DATA INPUT



TYPICAL COMMAND PROCESSING --

The "ERASE POINT" Service Routines

